PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/60

A1

(11) International Publication Number: **WO 00/13123**

(43) International Publication Date: 9 March 2000 (09.03.00)

(21) International Application Number: PCT/US99/19821

(22) International Filing Date: 27 August 1999 (27.08.99)

(30) Priority Data:
09/143,277 28 August 1998 (28.08.98) US

(71) Applicant: RECOVERY SALES CORPORATION [US/US]; 13900 East 35th Street, Independence, MO 64055 (US).

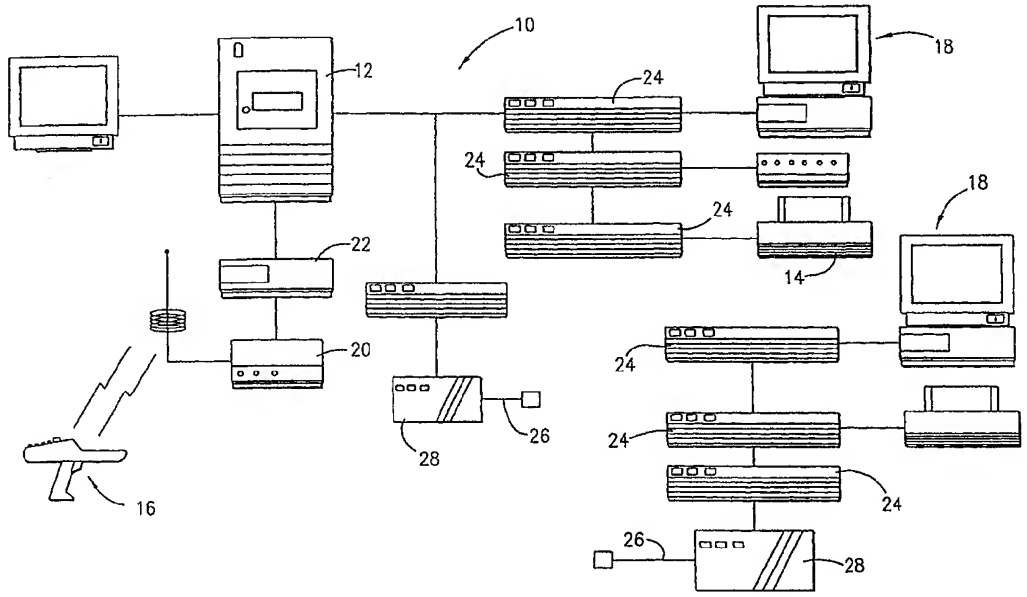
(72) Inventors: RAUBER, Brett, Alan; 3112 S. Vista Court, Independence, MO 64057 (US). PACK, Louis, D.; 4900 W. 83rd Terrace, Prairie Village, KS 66207 (US). BOGUE, Blaine, Everett; Apartment 6, 19008 E. 37th Terrace, Independence, MO 64057 (US). MYERS, Paul, David; 19935 Highway 273, Platte City, MO 64079 (US). HANSHAW, Frank, III; 18 David Drive, Sinsbury, CT 00670 (US). RUIZ, Robert, Anthony; Apartment #C39, 90 Farmington Avenue, Britain, CT 06053 (US).

(74) Agents: RUDY, William, A. et al.; Lathrop & Gage, LC, Suite 2800, 2345 Grand Boulevard, Kansas City, MO 64108 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: METHOD FOR MANAGING INVENTORY



(57) Abstract

A method and apparatus for managing inventory in a distressed inventory warehouse with the aid of a programmable computer is shown in the Figure. The inventory apparatus (10) includes a host computer (12), one or more printers (14) and a plurality of programmable input devices such as portable scanning devices (16) or remote computers (18). The method and apparatus provide for the complete automation of the inventory management of a distressed inventory warehouse. The method and apparatus also provides for the continual updating of records related to the distressed inventory as it passes through various stages within the distressed inventory warehouse.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

METHOD FOR MANAGING INVENTORY

Background of the Invention

1. Field of the Invention

The present invention relates to a method for managing inventory with the aid of a programmable computer, and more particularly to a method for receiving, sorting, marking, tracking, and selling distressed inventory which has been delivered to a distressed inventory central warehouse. The method provides for the complete automation of the inventory management of a distressed inventory warehouse and the continual updating of inventory data records related to the distressed inventory as it passes through various stages within the warehouse, including matching and identifying lost inventory..

2. Description of the Prior Art

Approximately 1% of all inventory shipped by freight carriers in the U.S. does not reach its intended destination or is not accepted by the buyer once it reaches its destination. This type of inventory is commonly referred to as "distressed inventory." Inventory often becomes distressed inventory because it was accidentally loaded on the wrong freight truck, damaged, or simply marked improperly.

Distressed inventory often cannot be feasibly reunited with the original inventory manufacturer or seller because freight carriers often do not immediately return to the place where they initially picked up the inventory. Therefore, it is common for freight carriers to deliver distressed inventory to centrally located distressed inventory warehouses that sort, store, redeliver or sell the inventory.

Once the distressed inventory is delivered to the distressed inventory warehouse, the distressed inventory warehouse must manage the distressed inventory and determine how to best deal with it. There are many unique problems associated with the managing of distressed inventory that are not encountered in the management of "regular" inventory. For example, regular inventory is typically delivered to a warehouse only after it has been ordered. Further, the ordered inventory will usually be clearly identified, typically in the form of documentation which accompanies the inventory (i.e., freight bills, labels, UPC code, etc). Yet further, the delivery dates and times for regular inventory is

set well in advance of the delivery. In contrast, with distressed inventory, it is impossible to anticipate what inventory will be lost, damaged, etc., during shipping; therefore, it is also impossible to anticipate what types of goods will be delivered to the distressed inventory warehouse and when the distressed inventory will be delivered. Accordingly, distressed inventory warehouses must be constantly prepared to receive and manage all types of inventory, particularly unidentified inventory, at all times.

Another unique problem associated with the management of distressed inventory is that many different types of inventory may be received by a single distressed inventory warehouse without advance notice. For example, it is not uncommon for distressed inventory warehouses to receive food items, clothing, vehicle parts, electronics, personal hygiene products, appliances, musical instruments, cleaning supplies and other similar goods. As can be appreciated, each of these types of inventory require different handling procedures. Thus, it is much more difficult to develop procedures for handling distressed inventory than for non-distressed inventory.

Another unique problem associated with the management of distressed inventory is that the distressed inventory is not always handled in conventional manners. For example, distressed inventory is often in such bad shape that it must be trashed or sold for scrap value. Additionally, some freight carriers require that certain types of distressed inventory be returned to them. Thus, distressed inventory warehouses must identify the distressed inventory that requires special handling and separate it from other types of inventory. Since different types of inventory must pass through different inventory management stages, it is exceedingly difficult to accurately manage and track all the distressed inventory received by a distressed inventory warehouse with known inventory tracking methods.

Another problem associated with the management of distressed inventory is that it is difficult to obtain and maintain sufficient records for use in pricing the distressed inventory for sale.

Methods and apparatuses for managing and tracking inventory are known in the art. However, none of these prior art methods and apparatus have been designed to take into account the unique difficulties associated with managing distressed inventory. Accordingly, there is a need for a method and apparatus for tracking inventory that is particularly configured for managing distressed inventory.

Objects and Summary of the Invention

In view of the problems associated with managing distressed inventory as set forth, it is an object of the present invention to provide a method and apparatus for managing and tracking inventory with the aid of a programmable computer that is particularly designed to take into account the unique difficulties associated with managing distressed inventory.

It is a more particular object of the present invention to provide a method and apparatus for managing and tracking distressed inventory with the aid of a programmable computer that allows an operator to easily and effectively identify and mark distressed inventory as it arrives at the distressed inventory warehouses.

It is another object of the present invention to provide a method and apparatus for managing and tracking distressed inventory with the aid of a programmable computer that allows an operator to easily and effectively track and sell inventory after it has been identified and marked and as it passes through various other stages in the distressed inventory warehouse.

It is another object of the present invention to provide a method and apparatus for managing and tracking inventory with the aid of a programmable computer that creates a unique inventory data record for each piece of inventory that is received by the distressed inventory warehouse and that permits the inventory data record to be instantaneously updated from anywhere in the distressed inventory warehouse to indicate the status of the inventory.

It is another object of the present invention to provide a method and apparatus for managing and tracking distressed inventory with the aid of a programmable computer that permits sales people to enter into negotiations with potential customers for the sale of the distressed inventory anywhere in the distressed inventory warehouse and that permits the appropriate inventory data records to be continually updated to indicate these negotiations.

It is another object of the present invention to provide a method of creating and managing a database of product UPC codes for use in pricing distressed inventory for sale.

In view of these and other objects that become evident from the following description of the present invention, a method and apparatus that is particularly designed for managing and tracking distressed inventory with the aid of a programmable computer is provided. The preferred inventory tracking apparatus broadly includes a centrally located

host computer, a printer, and a plurality of remote programmable input devices located throughout the distressed inventory warehouse and in communication with the host computer.

The host computer is operable for receiving inventory information relating to the distressed inventory and for creating and storing inventory data records containing the inventory information. The label printer is coupled with the host computer and is operable for printing machine-readable inventory labels that relate to their respective inventory data records for placement on the inventory.

The programmable input devices are in communication with the host computer and may include portable scanning devices, remote access terminals or computers or other types of programmable devices. The programmable input devices are operable for inputting or scanning the machine-readable inventory labels and for communicating with the host computer for receiving and transmitting tracking, sales and other status information.

The method of the present invention is performed with the aid of a computer program for operating the host computer described above. The method broadly includes the steps: a) receiving into the host computer inventory information related to a piece of inventory, the inventory information including inventory identification information and inventory destination information; b) storing the inventory information in an inventory data record in the computer; c) printing a machine-readable inventory label with a printer coupled with the computer, the inventory label including an inventory code associated with the inventory data record; d) placing the inventory label on the inventory; e) entering the inventory code into a programmable input device in communication with the computer for accessing the inventory data record; f) entering updating information into the programmable input device whenever the inventory information changes; g) sending the updating information from the programmable input device to the computer; and h) updating the inventory data record stored in the host computer in response to receipt of the updating information.

The updating information specified above may include order information indicating that a customer has offered to buy the inventory, tracking information indicating that the inventory has been moved to a new location within the warehouse, and pricing information to change the price of the inventory. This updating information is transmitted from the programmable input device to the host computer to update the inventory data record for constantly maintaining accurate inventory information within the host computer.

The present invention also includes a method for creating and managing a database of product UPC codes for use in pricing distressed inventory for sale.

Brief Description of the Drawing Figures

A preferred embodiment of the present invention is described in detail below with reference to the attached drawing figures, wherein:

Fig. 1 is a schematic representation of the inventory management apparatus constructed in accordance with a preferred embodiment of the present invention;

Fig. 2 is a flow diagram illustrating an overview of the steps in the preferred inventory management method of the present invention;

Fig. 3 is a flow diagram illustrating the inventory unloading and sorting portion or subroutine of the present invention;

Fig. 4 is a flow diagram illustrating the retail order and load-out portion or subroutine of the present invention;

Fig. 5 is a flow diagram illustrating the warehouse invoice and loadout process portion or subroutine of the present invention;

Fig. 6 is a flow diagram further illustrating the warehouse invoice and loadout process portion or subroutine of the present invention, and specially illustrating the hold and release portions of the warehouse subroutine;

Fig. 7 is a flow diagram further illustrating the warehouse invoice and loadout process portion or subroutine of the present invention, and specially illustrating the flag and unflag portions of the warehouse subroutine;

Fig. 8 is a flow diagram illustrating the bid sale and loadout portion or subroutine of the present invention;

Fig. 9 is a flow diagram illustrating the pricing and marking portions and subroutines of the present invention;

Fig. 10 is a flow diagram illustrating the price markdown and markup portion or subroutine of the present invention;

Fig. 11 is a flow diagram illustrating the overgoods management portion of the present invention; and

Fig. 12 is a flow diagram illustrating the search and match process of the overgoods management portion of the present invention.

Detailed Description of the Preferred Embodiments

I. INVENTORY MANAGEMENT APPARATUS

Turning now to the attached drawing figures, Fig. 1 illustrates the preferred inventory management apparatus 10 of the present invention. The following description of the inventory management apparatus 10 is provided for disclosing a preferred embodiment and best mode of the present invention. Those skilled in the art will appreciate that the preferred components of the inventory management apparatus 10 may be replaced with equivalent components without departing from the scope of the present invention.

The preferred inventory apparatus 10 broadly includes a host computer 12, one or more printers 14 coupled with the host computer 12, and a plurality of programmable input devices in communication with the host computer 12. The programmable input devices may include one or more remote computers 18, or any other types of input devices.

In more detail, the host computer 12 is provided for receiving inventory information relating to the distressed inventory and for creating and maintaining inventory data records containing the inventory information. The host computer 12 may be any type of minicomputer, microcomputer or mainframe computer but is preferably a Hewlett Packard 9000 Series 800 Model E server computer that uses the HP-Unix operating software and includes a conventional display monitor, a digital data storage tape drive, dual two-giga byte mirroring hard drives, and an integrated local-area-network (LAN) connection. Those skilled in the art will appreciate that the host computer 12 may be any type of computer including a microcomputer, minicomputer, or mainframe computer.

The printer or printers 14 are coupled with the host computer 12 and may be any conventional printer operable for printing machine-readable inventory labels. In preferred forms, a plurality of printers 14 are provided at various locations throughout the distressed inventory warehouse.

The programmable input devices are operable for scanning or entering the machine-readable inventory labels and for communicating with the host computer 12 to provide updating information to the host computer 12. As discussed above, the programmable input devices may include the portable scanning devices 16 or the remote computers 18. In preferred forms, a plurality of programmable input devices are provided at various locations throughout the distressed inventory warehouse.

The portable scanning devices 16 are preferably laser radio frequency (RF) terminals manufactured by Symbol Technologies. Each of the RF scanning devices 16

preferably includes a programmable microprocessor, a barcode scanner, an alphanumeric keypad, a portable printer, and a radio transceiver.

The RF scanning devices 16 communicate with the host computer 12 by transmitting and receiving radio signals to and from an RF transceiver 20 coupled with the host computer 12. The preferred RF transceiver 20 is also manufactured by Symbol.

A serial access bridge 22 also preferably manufactured by Symbol Technologies serves as a bridge between the RF transceiver 20 and the host computer 12. The serial access bridge 22 manages radio frequencies used for transmission, determines the most optimum frequencies in the surrounding environment, and determines the "chipping" sequence used to encrypt all transmissions between the RF scanning devices 16 and the host computer 12. The serial access bridge 22 also manages the transmission handoffs between the RF scanning devices 16 and the RF transceiver 20 by assigning unique configuration numbers to each of the RF scanning devices 16 and the RF transceiver 20 so that transmissions from each unit can be recognized.

Each remote computer 18 is preferably a conventional IBM compatible or equivalent microcomputer having a 486 or Pentium type microprocessor and a display screen; however, they may also be "dumb" terminals with input capabilities only. In preferred forms, a plurality of remote computers 18 are positioned in various strategic locations within the distressed inventory warehouse.

A plurality of data communications and terminal controllers 24 are provided for coupling the host computer 12 with the remote computers 18 and printers 14. The datacommunications and terminal controller 24 transfers data between the remote computers 18 and the host computer 12 and controls communications that would otherwise be controlled by the host computer 12, thus freeing the host computer 12 from these management functions. The datacommunications and terminal controllers 24 are preferably 16-port devices manufactured by Hewlett Packard and are connected with the host computer 12 by conventional LAN cabling.

As illustrated, the host computer 12 may also be coupled with remote computers 18 that are positioned outside the distressed inventory warehouse by a digital leased line 26. The digital leased line 26 is preferably a 4-wire conductor provided by an all-digital transmission network.

A pair of data/channel service units 28 provide for communication between the host computer 12 and the remote computers 18 positioned outside the distressed inventory warehouse. Each data/channel service unit 28 functions as a digital modem,

boosting signals across the digital leased line 26. Each data/channel service unit 28 is designed for direct connection to the digital leased line 26 and is operable for transmitting data at up to 56,000 bits-per-second (bps). The data/channel service units 28 are preferably manufactured by Motorola.

As will be recognized, any equipment with equipment or better capabilities, features, etc. can be used to practice this method.

II. INVENTORY MANAGEMENT METHOD

The method of the present invention manages distressed inventory or freight delivered to a distressed inventory warehouse. As used herein, inventory and freight are considered to be synonymous and are therefore used interchangeably. Additionally, as used herein, distressed inventory or freight is understood to include all types of inventory, particularly unidentified inventory, that is delivered to a distressed inventory warehouse by a freight carrier, including but not limited to, food items, automobile parts, clothing, personal hygiene products, art work, furniture, appliances, electronics, rugs, musical instruments, sports equipment, jewelry, and all other consumer wholesale and retail goods.

The inventory management method of the present invention is implemented with the aid of a computer program for operating the inventory management apparatus 10 described above. The computer program is preferably written in Pick Basic language, but may also be written in other conventional program languages. The computer program is preferably stored in the read-only memory (ROM) of the host computer 12, but may also be stored in the host computer's hard drive or in external disks or tapes for transfer to the memory of the host computer 12.

The source code for the computer program is reproduced in the appendix attached in the parent application. The flow diagrams in the drawing figures provide a high-level overview of the computer program.

Many of the method steps described and illustrated herein require an operator or distressed inventory warehouse salesperson to enter or scan in information concerning the inventory. These scanning or entering steps can be performed at any of the components of the inventory management apparatus 10 including the host computer 12, any of the RF scanners 16 or any of the remote computers 18. This allows the inventory to be managed from anywhere within the distressed inventory warehouse.

Fig. 2 illustrates an overview of the inventory management method of the present invention. As illustrated generally in step 200 of Fig. 2, inventory or freight is first

received by the distressed inventory warehouse and unloaded. Then, at step 202, the freight is sorted into several categories that determine how the freight will be managed within the distressed inventory warehouse. In preferred forms, the freight is sorted into the following categories: (1) freight that will be sold for scrap value (step 204); (2) freight that will be trashed (step 206) ; (3) over freight that will be reunited with the freight carrier or manufacturer (steps 208 and 210); (4) freight that will be stored, managed, and sold in the warehouse of the distressed warehouse inventory (steps 212, 214, 216, and 218); and (5) freight that will be stored, managed, and sold in a retail portion of the distressed inventory warehouse (steps 220, 222, 224, 226, and 228). The management, tracking, and sales of the freight during each of these categories or stages is discussed in more detail below.

UNLOAD PROCESS

Fig. 3 illustrates the freight unloading portion or subroutine of the present invention. In general, this portion or subroutine of the method provides for the unloading of the freight carrier, the identifying of the inventory delivered, and the creation and storage of inventory data records in the host computer 12 for use in managing and tracking the inventory as it passes through various stages within the distressed inventory warehouse.

In more detail, the inventory or freight is first delivered to the distressed inventory warehouse by a freight carrier. After the freight carrier has arrived, an operator logs the arrival time and unload start time of the freight carrier. Then, the operator initiates the unload subroutine or portion of inventory management method by accessing the appropriate subroutine in the host computer 12 as illustrated generally in step 300. Access may include conventional log-on or connection procedures and may provide an initial display screen informing the operator of the options available in the unload subroutine.

After the freight has been unloaded, it must be identified before proceeding through various stages in the distressed inventory warehouse. To identify the freight, the operator first determines whether the inventory has been delivered with a freight bill as illustrated in step 302.

If a freight bill has been provided with the inventory, the operator scans the bill or enters the freight bill information manually into the host computer 12 or one of the programmable input devices in communication with the host computer 12 in step 304. Alternatively, if a freight bill has not been provided with the inventory, the computer program prompts the operator to enter a description of the inventory including the

inventory type, quantity, condition and other information to describe the inventory in step 306.

Next, in step 308, the computer program prompts the operator to enter the location within the distressed inventory warehouse where the inventory will be initially stored before it proceeds through further stages of the inventory management method. As described in more detail below, this location may include a warehouse area, a retail area, or other specialized areas for particular types of freight.

After receiving this inventory identification information and location information, the host computer 12 stores the information in an inventory data record as illustrated generally in step 310. The host computer 12 creates a unique inventory data record for each piece of inventory received in the distressed inventory warehouse. Inventory labels are then printed by a printer 14 coupled with the host computer 12 or any one of the remote computers 18 and applied to the pieces of inventory as indicated in step 312. Each inventory label preferably includes a machine-readable code that identifies or is associated with its respective inventory data record.

During the unload portion or subroutine of the inventory management method, the operators also sort the inventory into groups and place groups of inventory onto portable storage containers such as pallets. Then, the host computer 12 creates and stores a portable grouping unit (PGU) data record for each pallet. Each PGU data record identifies each and every piece of inventory on its respective pallet. The host computer 12 also creates a pallet identification number, or PGU number, for each pallet. Finally, PGU labels that identify their respective PGU data records are printed and applied to the pallets. Once the unload portion or subroutine of the inventory management method is complete, the operator logs the unload end time, and the pallets of inventory are moved to their proper locations within the distressed inventory warehouse. As used herein, overfreight and overgoods are considered to be synonymous and are therefore used interchangeably.

OVERGOODS MANAGEMENT

Figure 11 illustrates the overgoods management portion or subroutine of the inventory management method. In general, the overgoods management portion of the method manages the distressed inventory in those situations in which the inventory is identified as overfreight/overgoods, and cannot yet be released to the other stages of the inventory management method. While the preferred embodiment described below is directed to managing certain distressed inventory, it is to be recognized by one of skill in

the art that management of any type of inventory is within the scope and spirit of the invention, even if the inventory is not categorizd as distressed inventory.

As inventory is received by the distressed inventory warehouse and unloaded, as illustrated in Figures 2 and 3, the inventory is sorted and moved to the proper location, as illustrated in steps 200, 202, and 314. As will be recognized by one of skill in the art, the term "warehouse" includes any storage facility. If the inventory meets predetermined criteria, which may be defined by each individual carrier, then the inventory is designated as overgoods inventory and moved to an area designated for receipt of overgoods inventory, and the overgoods process begins, step 1100. The predetermined criteria may be a code assigned to the shipment before its receipt, a dollar value for any particular shipment of inventory, a special account designation based upon the manufacturer of the item, or any other desired criteria. If the inventory does not meet this predetermined criteria, then it is not designated as overgoods inventory and is categorized appropriately, as illustrated on Figure 2 as discussed above.

When the overgoods inventory arrives in the area designated for overgoods, it may already have attached thereto the inventory label created in step 312. After the overgoods inventory arrives, an operator examines the shipment and enters overgoods information for each piece of overgoods inventory into a separate overgoods record in the computer, as illustrated in step 1102. The overgoods information includes descriptive product information, location information, and carrier information. The location information includes a corresponding PGU number associated with each piece of inventory. It is to be noted that a "piece" of inventory may be a single container or several related containers, such as an entire pallet of containers. The corresponding overgoods record may contain information for each individual container, or for each group of related containers.

As will be recognized by one with skill in the art, the descriptive product information may be made up of any information associated with the overgoods inventory. The descriptive product information in a preferred embodiment of the present invention includes designation of the inventory into one of five different classifications, each of which has different descriptive product information associated therewith. These classifications, with examples of detailed descriptive product information, are:

Clothing - RN number, style, manufacturer, color, size, UPC;

Long Iron - Ferrous, color code, type, manufacturer, weight, dimension, alloy code, UPC;

Tires - tire size, type (white wall, etc.), manufacturer, UPC;

Fabric - manufacturer, pattern, weave, material, size, weight, color, UPC;

General - UPC manufacturer, style, model, serial number, size, color, weight.

The information entered includes a cross-reference, generally by way of an identification number, with the inventory data record created in step 310, illustrated in Figure 3. This provides a cross-reference between the record generated during the unload process and the record generated in the overgoods process.

Once the descriptive product information has been entered, the operator may then perform a search of a shortage file or any outstanding inquiry file to attempt to match the inventory before it is moved to the overgoods storage area, as illustrated in step 1104. The shortage file includes information provided by the carriers regarding inventory that the carriers know is unaccounted for. This shortage information is not provided by all carriers, and may not be searched for every overgoods record entered into the computer. An outstanding inquiry file may include inquires related to certain distressed shipments recorded by carrier representatives. If there is a match when searching the shortage or inquiry file, then the inventory is removed from the overgoods area and is generally returned to the carrier, although it may also be reclassified in accordance with step 202, discussed above, but may be disposed of in other manners. An overgoods record is still generally created for this inventory.

If a match is not found corresponding to the shortage file, the operator may then generate a digital image of the overgoods inventory to be stored with the overgoods record corresponding to the overgoods inventory, as illustrated in step 1106. While this step is not performed for all overgoods inventory, it is often advantageous to store an associated image for inventory that is particularly difficult to describe. In a preferred embodiment, the operator focuses a digital camera on the overgoods inventory and captures the image from the digital camera to associate with the overgoods record. It is to be noted that one of skill in the art will recognize other manners in which to capture a digitized image to associate with the record other than the one described, all of which are within the spirit and scope of the present invention.

When the desired descriptive information has been entered into the overgoods record, the operator then enters location information into the record. The location information identifies the portable grouping unit (PGU) with which the inventory is located. The specific location of each PGU, and a specific location of the inventory, is then updated in each associated overgoods record when the PGU is moved to its proper location within the distressed inventory warehouse. This location can be input into the overgoods

record by any RF scanning unit 16 or remote computer 18 to update the location information of the PGU, and its corresponding inventory, whenever the PGU is placed or relocated within the distressed inventory warehouse. The location maintained in the corresponding overgoods record is the current location of the inventory.

The information entered into the overgoods record regarding the overgoods inventory may include designation of special status, as discussed in greater detail in the warehouse invoice and loadout process section below. Such special status will prohibit the item from ever being sold, and may require other special actions. Examples of the type of inventory that may be designated with special status information are prescription drugs, narcotics, alcohol and tobacco, high value items, firearms, and ammunition. Further, certain manufacturers and distributors require carriers to return distressed inventory, without release to a distressed inventory central warehouse.

After the overgoods record has been entered into the computer by an operator, the record is saved in the computer and the computer generates an overgoods record number, preferably a sequential number. Preferably, this number contains a code that relates the overgoods inventory to its associated carrier. A label is then printed by a printer 14 coupled with the computer 12 or any one of the remote computers 18, as illustrated in step 1108. The label is applied to the inventory and is indicated in step 1108. Generally, the label contains a bar code identifying the proper corresponding record, and may contain information, such as the overgoods record number and product description, that may be read by an operator. After the label is placed upon the inventory, the inventory is placed in or on the PGU. Once a PGU has reached the capacity of inventory it may contain, it is moved to a location within the distressed inventory warehouse, and the corresponding location information is updated.

The overgoods records in the computer may be searched for records matching preselected criteria, which may generate at least one matched overgoods record corresponding to matched overgoods inventory. The preselected criteria preferably includes information about the carrier associated with the overgoods inventory. This search process may be performed by an operator from the host computer 12, remote computers 18, or by personnel associated with the carriers who have a computer system connected to the host computer 12, such as by a wide area network. An operator must log into the system and provide a login code, such as a user name and password, before access will be granted, as illustrated in step 1202. Access by an operator may be, and preferably is, restricted to certain overgoods records. For example, personnel from carrier A will be

allowed access only to overgoods records associated with overgoods inventory received from carrier A, and not allowed access to overgoods records corresponding to overgoods inventory received from any other carrier. The specific login code will also indicate whether the operator may properly designate a record as a "match," as discussed in more detail below. Some personnel will be able to access the system and perform a search for several or all carriers simultaneously.

Once the operator has provided an appropriate login code and selects the appropriate subroutine to search the overgoods system, the computer will prompt the operator for search criteria which includes a text field where the operator may enter a string of alphanumeric characters to search. While it is to be recognized by one of skill in the art that the search criteria may be customized to fit the particular needs of the specific application, in a preferred embodiment the search criteria include: carrier; commodity code; quantity range (e.g. all items with the quantity being between five and ten); weight range (e.g. all items with weights between fifty pounds and seventy-five pounds); overgoods record date range (to search a date range during which the overgoods record was created); and shipping date range (to search date ranges during which the inventory may have been shipped). It is not necessary to enter values for each of the search criteria, but the search criteria are provided to limit the search results to more accurately locate records that may match the desired preselected criteria. The text field is provided to enter a string of alphanumeric characters to match characters that may appear in any field in the overgoods record. The alphanumeric string will also be checked against a file on the computer system containing synonyms of words, to improve the quality of the search. For example, if the alphanumeric string entered is "mitten," the system would also search for "mitt" and "glove." The text field will also accept wild card characters to search within a word or a part of a word. For example, the computer system would match an overgoods record containing the word "mitten" if the preselected criteria included "mit" with an appropriate wild card, such as *, for multiple letters. Single letter wild cards may also be provided. Further, a wild card character may be placed at the beginning of a word to find words having common endings. For example, if the alphanumeric string entered is "*mit," then the results would include records having the words, for example, "submit" or "summit."

Once the operator has entered the search criteria and any alphanumeric text in the text field, the operator may then command the computer to search the overgoods records for records matching the preselected criteria and may generate at least one matched

overgoods record. If no criteria are matched, then there will be no matched overgoods record generated. At that point, the operator may select to enter alternate criteria and alphanumeric text or to stop the search. Generally, the preselected criteria will include information about the carrier for the particular overgoods inventory. As a practical matter, this is primarily done through screening via the login code to allow access only to particular records corresponding to a particular carrier. Alternatively, carrier information may be entered into the search criteria.

Once the matched overgoods record is generated, a "results" screen is displayed to the operator to provide them with feedback regarding the search. A variety of information may be provided to the operator regarding the results of the search. For example, information may be provided regarding any synonyms that were matched, the total number of matches for all criteria, the number of matches for each search criteria or specific alphanumeric string presented, etc. The operator may then elect to view a summary of the search results which displays selected fields of the matched records to indicate what overgoods inventory was "matched," as illustrated in step 1206. The operator may further select to view all of the information corresponding to the matched overgoods record. Also, if an image has been entered for the particular overgoods record in step 1106, the user may elect to view that image in conjunction with the overgoods record.

If the operator is satisfied that the matched overgoods record is, in fact, a "match," the operator will designate the record as a match and input information into the "match" field associated with the matched overgoods record, as illustrated in step 1208. This action will also stamp the current record with the current login code and the date the match was found. The operator will also enter the appropriate shipping information for the necessary disposition of the item. If the operator does not have the proper authority to designate a record as a "match," then the record is designated "on hold," which also stamps the date and login code into the record. Periodically, preferably daily, a report is generated for all records designated "on hold," and those items are reviewed by one with proper authority to designate the record as a "match." It is to be noted that the entire quantity of the inventory associated with a matched record may not always be matched by the search criteria. For example, if the overgoods record indicates that the specific overgoods inventory is ten cartons of goods, and the search criteria only specifies five cartons of the goods, then a "match" may be indicated for only five of the cartons associated with the matched record.

If the matched overgoods record is designated as a "match" by carrier personnel conducting the search, then the carrier personnel knows of the matched overgoods record. If the matched overgoods record has been matched by other than carrier personnel, the carrier is notified of the matched overgoods record that matches the preselected criteria.

Once the "match" is designated by one with proper authority and the shipping information has been entered, the overgoods record is removed from the searchable database. If only a partial quantity from the record is "matched," then the record remains open in the searchable database for the remaining quantity.

After the item has been designated as a "match," and the associated shipping information has been entered, then a printer 14 associated with the computer system prints information regarding the matched overgoods record, as illustrated in step 1210. This information includes the current location of the inventory, which includes the current location of the associated PGU. Then the overgoods inventory is physically located in the overgoods storage area of the distressed inventory warehouse and relocated from its current location for further processing to appropriately dispose of the matched overgoods inventory, as illustrated in step 1212. Generally, the matched overgoods inventory is returned to the shipper with which it is associated or is forwarded to a consignee, but it is to be recognized by one of skill in the art that the matched overgoods inventory may be otherwise handled.

If the overgoods inventory associated with the matched overgoods record cannot be located in the overgoods storage area in the distressed inventory warehouse, an appropriate entry is made in the matched overgoods record such that if the record is ever accessed again, a message appears to the operator that this item was previously matched and not found. For example, if an item accidentally falls from one PGU into another PGU, then this inventory may not be found if it has been matched, because the search in the proper PGU would not locate an item. However, in the future, the item may be located in the incorrect PGU, and the record accessed to determine the status of the item. At that point, the message would appear to the operator that this inventory has been previously matched but was unable to be located. Supervisory personnel would then be notified to address the situation.

There will be some overgoods records that do not match any of the criteria entered during the search process. Some of these records will not match any of the criteria entered for many search processes. Part of the overgoods information initially received

into the overgoods record in the computer preferably includes a length of time that the inventory is to remain in the overgoods area of the distressed inventory warehouse, or a date on which the overgoods inventory may be released from the overgoods area of the distressed inventory warehouse. As a practical matter, these lengths of time are generally provided by the carrier. For example, carrier A may designate that all of the items designated as overgoods inventory received from that carrier must stay as overgoods inventory for at least five months, in order to be available to be matched and returned to the carrier. However, some inventory is not matched during that time period. Periodically, preferably each day, a report is printed that lists all overgoods inventory that has been designated as overgoods inventory for the preselected amount of time, or which has been designated for release on that particular day or earlier, but has not been released. These records are described as stale records. These stale records are included in a printout, along with corresponding location information for the overgoods inventory associated with the overgoods records that have not been matched and are due for release. The associated overgoods inventory is referred to as a matured item. The matured items are then physically relocated from their current location in the overgoods area of the distressed inventory warehouse for release to a salvage process. The salvage process is redesignation of the overgoods inventory for either scrap, step 204, trash, step 206, warehouse, step 212, or retail store, step 220, as illustrated on Figure 2. Then the distressed inventory will be managed as set forth in more detail herein.

In some cases, a carrier will have information regarding the shipper of the inventory, but not have sufficient information to deliver the inventory to the consignee or recipient. Generally, the carrier will then attempt to contact the shipper and determine the consignee of the shipment in question to enable appropriate delivery. However, if this fails, then the inventory that cannot be delivered may be called known overgoods, and is generally returned to the shipper, because the shipper information is known. The inventory management system of the present invention provides, either as a separate module or a subroutine of the present computer program, a known overgoods database for entry of information regarding these known overgoods.

When an item is determined by the carrier to be a known overgood, information regarding the known overgood is entered into the known overgood database in a computer, as in steps 306 and 1102. This information includes descriptive product information and shipper information. The descriptive product information may be the same as or different than that described above, depending upon the particular circumstances. It is

to be recognized by one of skill in the art that the specific descriptive product information and shipper information may vary from application to application without departing from the spirit or scope of the invention.

After the information has been input into the known overgoods database, the distressed inventory is generally returned to the shipper for appropriate disposition. The known overgoods database is available to be searched for records matching preselected criteria to help identify and locate where particular inventory may be, as in Figure 12. Often this preselected criteria is provided by the shipper or other requesting party to the carrier corresponding to requests regarding inventory that did not arrive at the intended destination. The search criteria and alphanumeric field of information may be similar to that which has already been described above.

If there are records in the known overgoods database that match the preselected criteria, then the shipper or other requesting party is provided the descriptive product information, which preferably includes the date the inventory was returned to the shipper for the inventory records that match the preselected criteria. The shipper may then physically identify the distressed inventory corresponding to the matched records by a review of the location of the inventory received from the carrier corresponding to the information provided from the matched record. Then the shipper may reship the identical inventory corresponding to the matched records or may select new inventory to be shipped to the consignee.

Often, a shipper, consignee, or other party will assert claims against a carrier for lost inventory. By use of the matching function of the known overgoods database, a carrier may provide information to the shipper or other party regarding inventory for which claims have been made that the carrier has actually returned to the shipper, forwarded to the consignee, or otherwise appropriately distributed. Thus, this matching function of the known overgoods database may prevent unnecessary and groundless claims against the carrier by shippers or other parties regarding inventory that has actually been returned to the shipper, forwarded to the consignee, or otherwise appropriately distributed.

RETAIL ORDER AND LOADOUT

Fig. 4 illustrates the retail order and loadout portion or subroutine of the inventory management method. In general, the retail order portion of the method manages distressed inventory in those situations involving a purchase by a customer of an item of

distressed inventory which cannot be taken through cashier lanes, or a purchase of distressed inventory at a reduced price.

In more detail, the retail order portion of the inventory management method is initiated in step 400 by an operator or salesperson after a customer has requested service. The computer program first prompts the operator to enter his or her salesperson I.D. number and the customer I.D. number in step 402. The computer program may also prompt entry of the customer's name, phone number, and other identifying information.

Next, in step 404, the computer program prompts for entry of inventory information identifying the inventory that the customer wishes to purchase. In preferred forms, the operator performs this step by scanning the inventory label placed on the inventory with one of the portable RF scanning devices 16 to access the inventory data records stored in the host computer 12.

After the inventory information is scanned or entered, the RF scanner 16 or remote computer 18 used for scanning or entering displays the assigned price for the inventory in step 406 and asks whether the price is acceptable to the customer. The inventory management method gives the salesperson the ability to either accept the assigned price for the distressed inventory or to perform a markdown of the price.

In the event of a markdown, the computer program prompts the sales person to enter the new price and a markdown code used to identify the reason for the markdown in step 408. This subroutine of the inventory management method also provides for the creation and storage of a data record containing a permissible markdown range for each salesperson. After the new price is entered, the host computer 12 compares the new price to the permissible markdown range for the corresponding sales person and verifies whether the markdown amount is within the permissible range. If the price is confirmed, the host computer 12 updates the appropriate inventory data record with this markdown information. The method also allows the operator to call up a pricing history for the inventory that lists the prices at which similar or identical pieces of inventory were sold to aid in the markdown pricing decision.

After the price of an item has been established and confirmed, the inventory management apparatus 10 creates a retail order for the sale of the inventory in step 410. The retail order may be generated by any of the RF units, remote computers 18, or host computer 12 and includes the name and password of the salesperson, the customer identification information, the inventory data record code or number, and the quantity of the inventory purchased.

As illustrated generally in steps 412, 414, 416, 418, 420, and 422, the retail order and loadout portion or subroutine also allows a salesperson to delete an order, change an order, and add more orders. The subroutine also allows a salesperson to view a running dollar total of items being purchased by a particular customer as the retail order is being processed and save and print the retail order and to go to another retail order for another customer. The salesperson can then later return to the prior customer's retail order.

The retail order and loadout portion of the method also allows a salesperson to view, at any time, on all RF units and all remote computers 18 or terminals, all items entered on any of the retail orders by entering the retail order's unique assigned number into any one of the RF units or remote computers 18.

After a retail order has been completed, it is saved in the host computer 12 as a retail order data record and is printed either on a dedicated printer coupled with the host computer 12 or on one of the portable printers 14 coupled with the RF scanning devices 16 or remote computers 18 as illustrated in step 424. The customer is then given a copy of the retail order to take to any cash register for payment.

To begin a retail checkout at a cash register, the cashier first presses a retail order button on the register and enters the retail order number in step 426. The portion of the method allows the cashier to adjust the tax status of each individual item if the buyer is tax-exempt. The method also allows the cashier to delete any items from the retail order and add any additional items that the customer has brought to the register not included on the retail order. The method then directs a printer 14 to print each item listed on the order on a point of sale (POS) receipt.

After the customer has paid for the inventory, a "pick" ticket is automatically printed at the distressed inventory warehouse loadout dock as illustrated in step 428. The pick ticket is used by the loadout person to locate the inventory and to transfer it to the dock door for loadout. The pick ticket may include, among other things, the retail order number, customer name, non-carryout items sold, quantity and location.

The method also generates a listing of non-carryout, purchased items on the loadout person's terminal for identifying items for customer pick-up. The listing of non-carryout, purchased items appears on a remote computer 18 positioned at the loadout dock when the loadout person enters an order number appearing on the retail order. The method permits the loadout person to scan out the purchased items as they are transferred to the

customer. The method causes an error message to appear on the loadout terminal in the event that a scanned item does not appear on the list of purchased items.

Finally, the retail order and loadout portion of the method prints a loadout confirmation after all non-carryout items have been scanned and transferred by the loadout person to the customer. The loadout person then has the customer sign the loadout confirmation to acknowledge receipt of the items. The host computer 12 stores the retail order information in step 430 and retains it for future pickups in cases where a customer cannot pick up all the items on the retail order.

WAREHOUSE INVOICE AND LOADOUT PROCESS

Fig. 5 illustrates the warehouse and loadout portion or subroutine of the inventory management method of the present invention. In general, this portion of the inventory management method provides for the invoicing, sales, and loadout of inventory stored in the warehouse portion of the distressed inventory warehouse.

In more detail, a salesperson initiates the warehouse invoice and loadout process by accessing the appropriate subroutine in the host computer 12 in step 500. The computer program then prompts the salesperson to enter his or her I.D. number and customer identification as illustrated in step 502. The program then determines if the customer has a current invoice on file. If a current invoice is found, it is displayed.

Step 504 lists the sale options for the inventory and prompts for a selection. In preferred forms, the method permits a salesperson to scan or enter a portable grouping unit (PGU) number, then either price the whole PGU, or scan in individual items on the PGU with individual prices and descriptions. The input of the PGU number can be performed by any of the RF scanning units 16 or remote computers 18.

The inventory management method provides for the following sales options:

(1) selling one PGU that only has a single item;
(2) selling one PGU that has many items;
(3) selling one item that has multiple PGU's;
(4) selling one PGU of one item with multiple PGU's;
(5) selling less than the whole amount of one item on a PGU;
(6) selling multiple items on multiple PGU's;
(7) selling one item of many items on a PGU; or
(8) selling all items except one on a PGU;

The subroutine also allows salespeople to delete or change the quantity, price, or description of any previously entered item or PGU.

After a sale has been made, the inventory management method creates a warehouse order and prints the order at any one of the printers 14 in step 506. The customer can then take this warehouse invoice to any cash register for purchase and loadout as described above. Step 508 then updates the appropriate inventory data records in the host computer 12. During loadout, the inventory management method requires the loadout person to sign for the sold items with a unique account number to ensure that the proper items were loaded out.

After a warehouse order or invoice has been created and saved, the warehouse portion or subroutine of the inventory management method allows a salesperson to call back the invoice to either add to it, make changes in it, or delete it. The method also allows the cashier to adjust a tax status of each individual item and allows the cashier to delete any items from the warehouse invoice and to add any additional items the customer has brought to the register not included on the warehouse invoice. Finally, once the inventory has been sold, paid for, and loaded out, the inventory management method indicates in the appropriate inventory data records that the purchased items are no longer in the distressed inventory warehouse.

To account for items which have been purchased but not picked-up, the inventory management method generates a report of purchased items that have not been picked up, identifying the items by customer name and phone number. This permits the distressed inventory warehouse to contact the customer. In the event that a customer purchases a whole PGU of multiple items and not all of the items are found at loadout, the inventory management method does not modify the inventory data records for the unfound items.

Figs. 6 and 7 are continuations of the warehouse invoice and loadout portion or subroutine of the inventory management method and illustrate the hold/release and flag/unflag options of method. These portions or subroutines permit any item or PGU entered on an invoice to be placed in a hold, release, flagged, or unflagged status.

For example, these subroutines permit a salesperson to input into the inventory management apparatus 10 a special status on any unsold item or PGU. The special status regarding a particular item is transmitted to the host computer 12 by way of any one of the RF scanning units 16 or remote computers 18 and is added to the appropriate inventory data records. The inventory data records can then be accessed by any RF

scanning unit 16 or remote computer 18 to obtain this special status information. The special status information is intended to alert any other salesperson of special information about that particular item, e.g., that a bid has already been placed on that item, that the item appears on a faxed bid offering, or that a buyer has made an appointment to see the item.

As illustrated generally in steps 600, 602, and 604 of Fig. 6 and steps 700, 702, and 704 of Fig. 7, the inventory management method prompts a salesperson wishing to add a special status to enter his or her identification, comments regarding the special status, and the date the special status was created. Once a special status has been entered for a particular piece of inventory, the inventory management method prohibits the sale of that piece of inventory. The inventory management method allows for the release of the special status to release the item for sale after a special release code has been entered into any of the portable RF units or remote computers 18.

BID SALE LOADOUT-PROCESS

Fig. 8 illustrates the bid sale and loadout portion or subroutine of the inventory management method. In general, this subroutine provides for the selling of inventory in the warehouse by a bidding or auction corresponds with the product category, and a reference price for which other stores such as Wal-Mart are selling the item.

After entering the above-described information, the operator is prompted to enter a selling price for the item in step 914. This selling price is saved with the child inventory data record for use in pricing other identical or similar types of inventory.

The inventory management method also provides for the creation and updating of a UPC code data base as illustrated generally in step 912. Each time an item's UPC code is scanned, the UPC information is saved in the UPC data base. The UPC data base also includes all of the information saved in the child data records described above. Thus, once a new UPC code is stored in the UPC data base, a permanent history of that product including detailed sales and markdown information is maintained. Then, each successive time that a UPC code is scanned for a product that has previously been scanned, the complete pricing history for that item can be retrieved and used for determining prices for the item.

PRICING MARKDOWN/MARKUP PROCESS

Fig. 10 illustrates the markdown/markup portion or subroutine of the inventory management method. In general, this subroutine provides for the marking down or marking up of the price or prices of an entire stock of a particular type of inventory that has been previously priced in the above-described pricing subroutine. This subroutine also allows for the marking down or marking up of the price or prices of selected portions of the stock of a particular type of inventory.

In more detail, the markdown/markup process is initiated when an operator accesses the appropriate subroutine of the inventory management method in step process, with the inventory typically being sold to the highest bidder.

In more detail, the bid sale and loadout process is begun when a salesperson accesses the appropriate subroutine in the inventory management method in step 800. To prepare the inventory for bid sale, it is first marked with bid lot numbers in step 802. Customers then submit bids for the inventory, and the bids are entered into any one of the RF scanning devices 16 or remote computers 18 in step 804.

Once all bids are received, the host computer 12 compares the bids and determines the highest bid price in step 806. The sales people then select the successful bidder, which is usually the high bidder. The inventory management method then creates, saves and prints a bid sale invoice in step 808. A copy of the bid sale invoice is given to the successful bidder in step 810 for loadout as described above. Finally, the appropriate inventory data records are updated and saved in step 812.

RETAIL PRICING AND MARKING

Fig. 9 illustrates the retail pricing and marking portion or subroutine of the inventory management method. In general, this subroutine provides for the separating or "exploding" of inventory into saleable units, pricing of inventory based on criteria determined from previous pricing steps, and marking of the inventory with new price labels or tags.

In more detail, this portion of the inventory management method is initiated when a sales person access the appropriate subroutine in the host computer 12 by way of a remote computer 18 in step 900. In preferred forms, a dedicated remote computer or computers are provided for these pricing and marking steps of the inventory management method.

As illustrated in step 902, salesperson or operator first sorts the inventory to separate out items that require special attention during pricing. For example, items of art, electronics, or building materials are often priced and managed differently than other types of inventory.

The operator then scans or enters the information on the inventory label in step 904 and review the information stored in the inventory data record. This allows the operator to determine such things as the quantity of that particular type of inventory that was received.

The operators may decide to divide up boxes of items into individual or package units that are more saleable in step 908. For example, distressed inventory warehouses often receive large boxes full of individual bars of soap. Since many consumers would rather purchase individual bars of soap rather than an entire carton, the inventory management method provides for the separating of the individual items into saleable units. This process is commonly referred to as "exploding".

After the inventory is exploded, a new "child" inventory data record is created with a number that relates to the original or "parent" inventory data record. The inventory management method then prompts the operators to fill in these new child data records with pricing information for the exploded items.

To assist the operator in pricing the individual saleable items, the items' UPC codes can be scanned or entered in step 910 to provide information which may be useful in pricing the item. The operator is also prompted to enter other information into the child inventory data record including the quantity and unit of measure of the item being priced, a detailed description of the item, the manufacturer of the product, a departmental code that 1000. The inventory management method then prompts the operator to scan the inventory label or enter the inventory numbers on the label for the items that are to be marked down or up. As described above, these inventory numbers can be entered in either the RF scanning devices 16 or the remote computers 18. The inventory management method then retrieves the appropriate inventory data record and displays on the terminal of the RF scanner or remote computer 18 the date the item was originally priced, the quantity of the item currently in the distressed inventory warehouse, the original price of the items, the product description, the product manufacturer, the most recent markdown/markup date if any, and the most recent markdown/markup price if any.

All of the above-described information can be used by the operator for determining a new price for the item. For example, if the operator learns that a large quantity of the

item is currently in the distressed inventory warehouse and that recent markdown prices have failed to increase sales of the item, he or she can more drastically reduce the price to increase sales.

After the operator enters the inventory numbers for the inventory that is to be marked up or marked down, the inventory management method prompts the operator to enter a new price in step 1002. The inventory management method then prompts the operator to input a reason code for the adjustment in step 1004 and modifies and updates the appropriate inventory data record to reflect the new price in step 1006. Finally, new price tags with the new updated prices are printed and applied to the inventory in step 1008.

PREFERRED EMBODIMENT OF SOURCE CODE

```
AVSU.L.MATCHINVEN.1
SUBROUTINE AVSU.L.MATCHINVEN.1(AV)
* Purpose :Postsub for matching
* Status :6.00.Y 09-03-97 20:53:35 D919FE1B
* Notes :A) 01-10-97 RAR initial version> >:B) 01-11-97 RAR continu> >:C) 01-13-97
RAR continue> >:D) 01-24-97 BAR change sleep inline 134 from 35 to 15> >:E)
05-12-97 RAR development> >:F) 06-06-97 RAR recompile> >:G) 06-09-97 FWN add
section to just view image> >:H) 07-01-97 FWN move to live> >:I) 07-02-97 FWN
debugging> >:J) 07-03-97 RAR remove temp zzz> >:K) 07-09-97 RAR use dummy
image to
cleanse Termite memory of last image saved> >:L) 07-09-97 FWN create code to make a
new subdirectory using the carrier id as the subdir name> >:M) 07-14-97 FWN correct
image problem - snappy stops after making a new directory> >:N) 07-15-97 FWN
continued chances> >:O) 07-18-97 FWN created new search feature and testing> >:P)
07-21-97 FWN changed the manner in which the new subdirectories are created> >:Q)
07-24-97 RAR recompile> >:R) 08-07-97 RAR Rearrange include file> >:S) 08-07-97
FWN added new search facility> >:T) t8-14-97 RAR continue> >:U) 08-15-97 RAR
continue> >:V) 08-19-97 RAR add update of manufacturer> >:W) 08-27-97 RAR change
to command,,X) 09-02-97 RAR subdirectory work> >:Y) 09-03-97 RAR conitnue; *
Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV
AVSU.N.MATCHINVEN; INCLUDE AV.EQUATE MATCHING. INVENTORY;
INCLUDE AVMP AVAL.EXEC.LEVEL; PORT=AV<1,1,1>;
WSID=PORT:'+':EXEC.LEVEL+1; DIRECTORY.TIME.LIMIT=30; NO.IMAGE=1;
```

```
TERMITE.OK=0; ACTION.LIST=''; PROMPT.LIST=''; HELP.LIST='';
ACTION.LIST.2=''; PROMPT.LIST.2= ''; HELP.LIST.2= ''; IF MODE#'NEW' THEN;
C.IMAGEID=ID[3,98765432];
SUBDIRECTORY=ITEM<MATCHINVEN.CARRIER>:'\'; END ELSE;
SUBDIRECTORY= ''; END; IF MODE='VIEW' THEN; GOSUB 100; * Verify that
application is being run from Termite; IF TERMITE.OK THEN; GOSUB 200; * Check to
see if image exists; IF NOT(NO.IMAGE) THEN;
ACTION.LIST<1,-1>='VIEW.IMAGE'; PROMPT.LIST<1,-1>=' View image';
HELP.LIST<1,-1>='View image';
BASEAVMENUID='AVUP.CHOOSE.TEMPLATE. PORT';
AVMENUID='AVUP.CHOOSE.':PORT:'.':EXEC.LEVEL; HEAD.TEXT='Select
option':VM; CALL
AVSS.MENU.FILL(AV,BASEAVMENUID,AVMENUID,HEAD.TEXT,PROMPT.LIST,
ACTION.LIST,HELP.LIST); LOOP; REPAINT='N'; CALL
AVSS.MENU(AV,'1',BASEAVMENUID,TI,REPAINT,'ON',SCREENPAINT); IF
REPAINT#'N' THEN REPAINT.LINES<-1>=REPAINT; UNTIL TI=ENT DO;
BEGIN CASE; CASE TI='VIEW.IMAGE'; GOSUB 300; * View Snappy image; GOTO
99; END CASE; REPEAT; END ELSE; CALL AVSS.DISPLAY.MESSAGE(AV,'No
image is on file'," ); END; END; END ELSE; IF
CURRATT=MATCHINVEN.MANUFACTURER AND
ITEM<CURRATT,CURRVAL>='' THEN; ACTION.LIST<1,-1>='ADD.MFR';
PROMPT.LIST<1,-1>=' Add manufacturer'; HELP.LIST<1,-1>='ADD.MFR';
END; ACTION.LIST<1,-1>='SNAPPY'; PROMPT.LIST<1,-1>=' Image editing';
HELP.LIST<1,-1>='SNAPPY'; ACTION.LIST<1,-1>='SEARCH';
PROMPT.LIST<1,-1>=' Search inventory'; HELP.LIST<1,-1>='SEARCH';
BASEAVMENUID='AVVP.CHOOSE.TEMPLATE.PORT';
AVMENUID='AVUP.CHOOSE.':PORT:'.':EXEC.LEVEL; HEAD.TEXT='Select
option':VM; CALL
AVSS.MENU.FILL(AV,BASEAVMENUID,AVMENUID,HEAD.TEXT,PROMPT.LIST,
ACTION.LIST,HELP.LIST); LOOP; REPAINT='N'; CALL
AVSS.MENU(AV,'1',BASEAVMENUID,TI,REPAINT,'ON',SCREEN PAINT); IF
REPAINT#'N' THEN REPAINT.LINES<-1>=REPAINT; UNTIL TI='EXIT' DO;
BEGIN CASE; CASE TI='ADD.MFR';
WSID='MATCHING.':AV<1,1,1>:'.':EXEC.LEVEL+1; WSITEM='MATCHING';
```

CALL AVSS.WORKSPACE(AV,WSID,WSITEM,'ON'); EXECTEXT='AV UP MFR NEW'; CALL AVSS.EXECUTE(AV,EXECTEXT,'N',",");
WSID='MATCHING.':AV<1,1,1>:'.':EXEC.LEVEL+1; CALL AVSS.WORKSPACE(AV,WSID,WSITEM,'OFF'); IF WSITEM#" AND WSITEM#'MATCHING' THEN; ITEM<CURRATT,CURRVAL>=WSITEM; CALL AVUP.FILL(AV,CURRATT,",CURRVAL,");
REPAINT.LINES<-1>=TM<1,1,1>:VM:BM<1,1,1>; CALL AVUP.REPAINT(AV); END; GOTO 99; CASE TI='SEARCH'; CALL AVSU.Z.MATCHING.FIND(AV,MATCHINVEN.CARRIER,C.COMMODITY.CODE,C.MATCHSEDN,
C.MATCHSEFN,'MATCHING.SEARCH',MATCHINVEN.UNMATCHED.SEARCH,C.MATCHSEFIDN,; C.MATCHSEFIFN); * DISPLAY.MESSAGE='This feature is currently under development.'; * CALL AVSU.Z.ALL.SEARCH(AV,C.MATCHSHFIFN,C.MATCHSHFIDN,C.MATCHSHFN, C.FINDATTLIST,; C.FIND.DEPTH); * PROBLEM.FOUND=1; ***; *** New version!!!; ***; SEARCH.STRING=ITEM<MATCHINVEN.PRODUCT>:'
':ITEM<MATCHINVEN.STYLE>:' ':ITEM<MATCHINVEN.MANUFACTURER>:'
'ITEM<MATCHINVEN.MODEL>:' ':ITEM<MATCHINVEN.RN.NUMBER>:'
':ITEM<MATCHINVEN.COLOR>:'
':ITEM<MATCHINVEN.SHIPPER.INFORMATION>:"
:ITEM<MATCHINVEN.CONSIGNEE.INFORMATION>:'
':ITEM<MATCHINVEN.ALLOY>:' ':ITEM<MATCHINVEN.PATTERN>:"
':ITEM<MATCHINVEN.WEAVE>:' ':ITEM<MATCHINVEN.MATERIAL>:'
':ITEM<MATCHINVEN.SIZE>; SEARCH.STRING=TRIM(SEARCH.STRING); *' The WSITEM is built in the following manner: <1> is the search; *' to be performed, <2> is the search string, <3> is the quantity,; * <4> is the weight, and <5> is the Carrier.commodity code.; WSITEM="; WSITEM<1>='MATCHING.SHORTAGE';
WSITEM<2>=SEARCH.STRING;
WSITEM<3>=ITEM<MATCHINVEN.QUANTITY>;
WSITEM<4>=ITEM<MATCHINVEN.WEIGHT>;
WSITEM<5>=ITEM<MATCHINVEN.CARRIER>; CALL AVSS.WORKSPACE(AV,WSID,WSITEM,'ON'); CALL AVSS.EXECUTE(AV,'AV UP MATCHING.SEARCH.STANDALONE NEW','N',;
REPAINT.LINES<-1>='3':VM:'24'; CALL AVUP.REPAINT(AV); CASE

T1='SNAPPY'; GOSUB 100; * Verify that application is being run from Termite; IF TERMITE.OK THEN; * If image has been scanned this session then or mode#new then; * Check drive to see if image is available from here.; IF C.IMAGEID#" THEN; * Check for C.temp.image; GOSUB 200; * Check to see if image exists; *** PROBLEM: We need to verify the existence of the temp; *** image if this is a "new" case and an image has previously; *** been created!!!; *** Check for no.image because it does not matter which Id; *** exists, provided that one of them triggers the flag to; *** indicate an image exists.; IF NO.IMAGE THEN; CODE=0; CALL PIX.EXISTS(C.IMAGE.PATH:C.TEMP.IMAGE,CODE); IF CODE=1 THEN NO.IMAGE=0; END; END; * Prompt to capture image (First time); IF NO.IMAGE THEN; IF C.MATCHING.EDIT THEN; ACTION='CAPTURE.NEW.IMAGE'; PROMPT=' Capture image'; LOCATE(ACTION,ACTION.LIST.2,1;LOCVAL;"AL") ELSE; PROMPT.LIST.2=INSERT(PROMPT.LIST.2,1,LOCVAL;PROMPT); ACTION.LIST.2=INSERT(ACTION.LIST.2,1,LOCVAL;ACTION); END; END; END ELSE; * View existing image; AVSU.W.MATCHINVEN; SUBROUTINE AVSU.W.MATCHINVEN(AV); * Purpose :; * Status :5.86.T 09-04-97 08:20:57 5DC724F1; * Notes :A) 04-22-97 FWN compile> >:B) 05-12-97 RAR development> >:C) 07-01-97 FWN move to live> >:D) 07-03-97 FWN input zzz for testing> >:E) 07-09-97 FWN added code to change the path name for saving/renaming images and verify that the new directory & subdirectory exists> >:F) 07-14-97 FWN minor change to "saving" logic> >:G) 07-15-97 FWN more changes> >:H) 07-21-97 FWN removed input zzz that was used by Bret during his demo> >:I) 07-28-97 FWN added code to unlock inventory record> >:J) 07-30-97 FWN changed Id fromeight chrs to nine chcs> >:K) 07-31-97 FWN added IDTYPE> >:L) 07-31-97 FWN added code to create the special description field> >:M) 08-02-97 FWN moved inventory unlocking record section to the Backend of Matching~inventory> >:N) 08-02-97 FWN added code that verifies the total amount of PGU quality has not been changed> >:O) 08-07-97 RAR Rearrange include file> >:P) 08-18-97 BAR add fill of receive.date> >:Q) 08-27-97 RAR use windows version of command com> >:R) 09-02-97 RAR directory works> >:S) 09-03-97 RAR continue> >:T) 09-04-97 RAR recompile; * Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV AVSU.N.MATCHINVEN; INCLUDE AV.EQUATE MATCHING. INVENTORY; INCLUDE AVMP AVAL.EXEC.LEVEL; PGULIST=' '; * Build PGULIST to compare changes from ORIGPGULIST.; PGULIST<1>=ITEM<MATCHINVEN.PORTABLE.GROUP>;

PGULIST<2> =ITEM<MATCHINVEN.PORTABLE.GROUP.QOH>; PGUL I
ST<3> =I TEM<MATCHINVEN . PORTABLE .GROUP.QTY . COMMITTED>;
C.PGULIST=PGULIST; * Before continuing, verify that the portable group total quantity;
* amount has not been changed!; TOTORIG=SUMMATION(C.ORIGPGULIST<Z>,;
TOTCURR=SUMMATION(C.PGULIST<2>); IF TOTORIG#TOTCURR THEN;
DISPLAY.MESSAGE='The total original quantity on hand (':TOTORIG:') does not equal
the total now showing on hand (':TOTCURR').'; PROBLEM.FOUND=1; GOTO 99;
END; ** * ** * ** * * ********** ********** * ***** *
********** *** *; *** Any error traping needs to be performed before this point ***;
**** ** * ******** **** ** **** ********** ***** ** **** ***;
* Stamp carrier commodity code; IF MODE='NEW' THEN;
ITEM<MATCHINVEN.CARRIER.COMMODITY> =ITEM<MATCHINVEN.CARRI
ER>:'+':C.COMMODITY.CODE;
ITEM<MATCHINVEN.COMMODITY.CODE> =C.COMMODITY.CODE; END; *
Stamp Receive date; IF ITEM<MATCHINVEN.RECEIVE.DATE> =" THEN
ITEM<MATCHINVEN.RECEIVE.DATE> =DATE(); * Stamp Mature.Date; * Restamp
commodity code from origitem with carrier code from item.; ; * This code is used when a
record is edited from the Matching.inventory; * dictionary.; IF
ORIGITEM<MATCHINVEN.CARRIER>#ITEM<MATCHINVEN.CARRIER> AND
ORIGITEM<MATCHINVEN.CARRIER>#" THEN;
ITEM<MATCHINVEN.CARRIER.COMMODITY> =ITEM<MATCHINVEN.CARRI
ER>:'+':FIELD(ITEM<MATCHINVEN.CARRIER.COMMODITY>,'+',2); END; *
Assign nextitem counter manually - To support multiple dicts; IF MODE='NEW' THEN;
* Use code from AVSS.NEXTITEM to get next available item based; * on carrier because
of funky multiple dictionary method; IDTYPE='P'; IDLENGTH='7'; IF
CURRNEXTITEM= " AND MODE='NEW' THEN;
CURRIDPREFIX=ITEM<MATCHINVEN.CARRIER>;
NIID='*NI.':CURRIDPREFIX; READVU NEXTITEM FROM
C.MATCHINVENDN,NIID,1 ELSE NEXTITEM=1;
CURRNEXTITEM=CURRIDPREFIX:STR('O',IDLENGTH-LEN(NEXTITEM)):NEXTI
TEM; READ TEMPITEM FROM FN,CURRNEXTITEM THEN; TEMPITEM= " ; *
AvCompiler; RELEASE C.MATCHINVENDN,NIID; CALL
AVSS.ERROR(AV,'FATAL','Matching Inventory number ':CURRNEXTITEM:' has
already been issued !!!!'); STOP; END; NEXTITEM=NEXTITEM+1; WRITEV

```
NEXTITEM ON C.MATCHINVENDN,NIID,1; ID=CURRNEXTITEM; END; END; *
Rename Image; *** This code belongs above the line!!!; IF C.RENAME.IMAGE THEN; *
Verify that the directory to store the images exists!!!; DIRECTORY.FOUND=0;
SUBDIRECTORY=ITEM<MATCHINVEN.CARRIER>:'\'; CALL
PIX.EXISTS(C.IMAGE.PATH:SUBDIRECTORY,DIRECTORY.FOUND); IF
NOT(DIRECTORY.FOUND) THEN; COMMAND='COMMAND /C MKDIR
':C.IMAGE.PATH:ITEM<MATCHINVEN.CARRIER>; CALL
PIX.CALL.DOS(COMMAND,1); SLEEP 2; CALL
PIX.EXISTS(IMAGE.PATH,DIRECTORY.FOUND); IF NOT(DIRECTORY.FOUND)
THEN; DISPLAY.MESSAGE='The directory ':IMAGE.PATH:' does not exist. Please
contact your system administrator for further asistance.'; GOTO 99; END; END;
```

* Copy temp image to permanent image ID; TEMPNAME=C.TEMP.IMAGE; C.IMAGEID=ID[3,987654321]; COMMAND='COMMAND /C COPY ':C.IMAGE.PATH:TEMPNAME:' ':C.IMAGE.PATH:SUBDIRECTORY:C.IMAGEID:'.BMP'; COMMAND=COMMAND; PRINT NOTEPRINT:'Renaming Image...'; CALL PIX.CALL.DOS(COMMAND,1); SLEEP 5; NOTE=''; PRINT NOTEPRINT:NOTE; * Check to see if image still exists; NO.IMAGE=1; CODE=0; ***; *** For some unexplainable reason, the image is not being saved on; *** Bret's system. Also, the wrapup is deleting the old code while; *** at the same not saving the new image!!! I believe this problem; *** is due to the fact that when two TERMiTE/Pick commands are issued; *** in succession, the latter command does not acknowlede the former.; ***; CALL PIX.EXISTS(C.IMAGE.PATH:TEMPNAME,CODE); IF CODE=1 THEN NO.IMAGE=0; IF NOT(NO.IMAGE) THEN; CALL PIX.ERASE.DOS.FILEC.IMAGE.PATH:TEMPNAME); PRINT NOTEPRINT:'Image Renamed !'; SLEEP 1; NOTE=''; PRINT NOTEPRINT:NOTE; END; END; * This section will create the special description field to help; * identify products when displayed in the Matching.search.standalone; * file.; COMMID=ITEM<MATCHINVEN.COMMODITY.CODE>; CALL AVSU.Z.MATCHSESA.DESCRIPTION(AV,COMMID,ITEM,DESCRIPTION); ITEM<MATCHINVEN.DESCRIPTION>=DESCRIPTION; 99*; RETURN; AVSU.L.MATCHSESA.2; SUBROUTINE AVSU.L.MATCHSESA.2(AV); * Purpose :Postsub for matching; * Status :6.00.H 08-14-97 23:16:01 50289263; * Notes :A) 04-21-97 FWN created>>:B) 04-22-97 FWN changes to new modifications>>:C) 06-03-97 RAR continue>>:D) 06-06-97 RAR CONTINUE,,:E) 06-06-97 FWN created new version>>:F) 07-01-97 FWN move to live>>:G) 08-13-97 RAR change to generic format for addition of matching.shortage>:H) 08-14-97 RAR use generic file method; Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV AVSU.C.MATCHSESA; INCLUDE AV.EQUATE MATCHING. INVENTORY; INCLUDE AV. EQUATE MATCHING . COMMODI TY;

; INCLUDE AV.EQUATE MATCHING.SEARCH.STANDALONE; INCLUDE AV.EQUATE MATCH ING . SHORTAGE; INCLUDE AVMP AVAL.EXEC.LEVEL; TARGID=ITEM<MATCHSESA.INVENTORY,CURRVAL>; IF TARGID=" THEN GOTO 99; BEGIN CASE; CASE C.FILE.TO.USE='MATCHING.SHORTAGE'; TARG.COMMODITY.CODE=MATCHSH.COMMODITY.CODE; CASE C.FILE.TO.USE='MATCHING.INVENTORY'; TARG.COMMODITY.CODE=MATCHINVEN.COMMODITY.CODE; END CASE; OPEN 'MATCHING.COMMODITY' TO MCOMMFN ELSE CALL AVSS.ERROR(AV,'NOFILE','MATCHING.COMMODITY'); READY DICTID FROM C.MATCHFILEFN,TARGID,TARG.COMMODITY.CODE ELSE DICTID= "; READY MATCHING.DICTIONARY FROM MCOMMFN,DICTID,MCOMM.INVENTORY.DICT ELSE MATCHING.DICTIONARY="; WSID=AV<1,1,1>:'+':EXEC.LEVEL+1; WSITEM='LOOKUP'; CALL AVSS.WORKSPACE(AV,WSID,WSITEM,'ON'); BEGIN CASE; CASE C.FILE.TO.USE='MATCHING.INVENTORY'; EXECTEXT='AV UP ':MATCHING.DICTIONARY:':':C.FILE.TO.USE:' VIEW ':TARGID; CASE C.FILE.TO.USE='MATCHING.SHORTAGE'; EXECTEXT='AV UP ':C.FILE.TO.USE:' VIEW ':TARGID; END CASE; CALL AVSS.EXECUTE(AV,EXECTEXT,'N','','' '; 99*; RETURN; AVSU.L.MATCHSESA.1; SUBROUTINE AVSU.L.MATCHSESA.1(AV); * Purpose :Postsub for matching; * Status :6.00.S 08-17-97 20:41:13 B615EF6C; * Notes :A) 04-21-97 FWN created>>:B) 04-22-97 FWN changes to new modifications>>:C) 06-03-97 RAR continue>>:D) 06-06-97 RAR CONTINUE>>:E) 06-09-97 FWN corrected error - when Searchstring blank routine blew-up>>:F) 06-10-97 RAR add sub 100>>:G) 06-11-97 FWN testing the "~" error>>:H) 06-12-97 FWN allow only one wild card per search>>:!) 06-13-97 FWN finish dup word check and test other commodity codes>>:J) 06-16-97 FWN correct the methodology of reading from the synonym file>>:K) 06-17-97 FWN changed the search function so that AVSS.FIND is only executed when Search.text is changed>>:L) 06-20-97 FWN added new fields code has to accomodate new field names>>:M) 06-23-97 FWN cleaned up display text>>:N) 06-25-97 FWN changed the order of options in menu>>:O) 07-01-97FWN compile - move to live>>:P) 07-31-97 FWN added section to build special description in the match standalone process,,:Q) 07-; 31-97 FWN remove special description code - now done in wrapup of Matching.inventory>>:R) 08-14-97 RAR change to use generic files instead of matchinven>>:S) 08-17-97 RAR

remove items with close.date; * Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV AVSU.C.MATCHSESA; INCLUDE AV.EQUATE MATCHING. INVENTORY; INCLUDE AV.EQUATE MATCHING.SEARCH.STANDALONE; INCLUDE AV.EQUATE MATCHING. SHORTAGE; INCLUDE AV.EQUATE MATCHING. SYNONYM; INCLUDE AVMP AVAL.EXEC.LEVEL; PORT=AV<1,1,1>; ; * Special !!!!!!; * Must set problem.found manually because this release of AVGEN does not; * expect to get problem.found called from a looksub; PROBLEM.FOUND=0; GROUPS=C.PAIRS; CALL AVSU.Z.ALL.SETS(AV,GROUPS); IF PROBLEM.FOUND THEN GOTO 99; * Verify that the value in the "To" position is less than the value In; * the "From" position.; IF ITEM<MATCHSESA.QUANTITY.TO> <ITEM<MATCHSESA.QUANTITY.FROM > THEN PROBLEM.FOUND=1; IF ITEM<MATCHSESA.WEIGHT .TO> <ITEM<MATCHSESA.WEIGHT . FROM> THEN PROBLEM. FOUND=1; IF ITEM<MATCHSESA.PRO.NUMBER.DATE.TO> <ITEM<MATCHSESA.PRO.NUMBER.DATE.FROM> THEN PROBLEM.FOUND=1; IF ITEM<MATCHSESA.SHORTAGE.RECORD.CREATE.TO> <ITEM<MATCHSESA.SHORTAGE.RECORD.CREATE.FROM> THEN PROBLEM.FOUND=1; IF PROBLEM.FOUND THEN DISPLAY.MESSAGE='The "From" value must be less than the "To" value'; IF PROBLEM.FOUND THEN GOTO 99; ACT ION . LIST=''; PROMPT.LIST=''; HELP. LIST=''; SEARCHSTRING=''; NUMTEXTLINES=DCOUNT(ITEM<MATCHSESA.SEARCH.TEXT>,VM); FOR 1=1 TO NUMTEXTLINES; TEXTLINE=ITEM<MATCHSESA.SEARCH.TEXT,I>; TEXTLINE=TRIM(TEXTLINE); ITEM<MATCHSESA.SEARCH.TEXT,I> =TEXTLINE; IF SEARCHSTRING='' THEN; SEARCHSTRING=TEXTLINE; END ELSE; SEARCHSTRING=SEARCHSTRING:' ':TEXTLINE; END; NEXT I; SEARCHSTRING=TRIM(SEARCHSTRING); WILD.CARD.COUNT=COUNT(SEARCHSTRING,'~'); IF WILD.CARD.COUNT>1 THEN; DISPLAY.MESSAGE='Only one "-" is allowed per search'; GOTO 99; END; * Check words and add synonyms to searchstr~ng; TEMPSTRING=SEARCHSTRING; GOSUB 300; * Create temp string to eliminate any special characters; NUMWORDS=DCOUNT(TEMPSTRING,,); FOR I=1 TO NUMWORDS; WORD=FIELD(TEMPSTRING,' ',I); READY SYNONYM FROM C.MATCHSYNFN,WORD,MATCHSYN.SYNONYM THEN; CALL

AVSS.REPLACE.STRING(AV,SYNONYM,VM,'|'); WORD.LEN=LEN(WORD); STRING.LEN=LEN(SEARCHSTRING); WORD.POS=INDEX(SEARCHSTRING,WORD,1); INSERT.POS=WORD.POS+WORD.LEN-1; SEARCHSTRING=SEARCHSTRING[1,1NSERT.POS]:'|':SYNONYM:SEARCHSTRIN GtlNSERT.POS+1,STRING.LEN]; END; NEXT I; * Check for dup words; DUPLIST="; TEMPSTRING=SEARCHSTRING; GOSUB 300; * Create temp string to eliminate any special characters; NUMWORDS=DCOUNT(TEMPSTRING,' '); FOR 1=1 TO NUMWORDS; WORD=FIELD(TEMPSTRING,' ', I); WORD.COUNT=1; J=0; LOOP; J=J+1; UNTIL J>NUMWORDS DO; IF J#l THEN; COMPARE.WORD=FIELD(tempstring,' ',J) NU,; IF COMPARE.WORD=WORD THEN WORD.COUNT=WORD.COUNT+4; END; REPEAT; IF WORD.COUNT>1 THEN; DUPLIST<1,-1>=WORD; END; NEXT I; IF DUPLIST#" THEN; DlSPLAY.MESSAGE='Duplicate words are not allowed'; GOTO 99; END; IF SEARCHSTRING#" THEN; ACTION.LIST<1,-1>='SEARCH'; PROMPT.LIST<1,-1>=' Search inventory'; HELP.LIST<1, -1,= ~ SEARCH'; END ELSE; DlSPLAY.MESSAGE='Enter search text before running a search,; END; IF C.LASTSEARCHITEM#" THEN; ACTION.LIST<1,-1>='RETURN'; PROMPT.LIST<1,-1>=' Return to previous search'; HELP.LIST<1,-1>='RETURN'; END ELSE; GOSUB 400; * Compare the last search with current search if same end; END; IF PROMPT.LIST=" THEN GOTO 99; TEXT="; BASEAVMENUID='AVUP.CHOOSE.TEMPLATE.PORT'; AVMENUID='AVUP.CHOOSE.':PORT:'.':EXEC.LEVEL; HEAD.TEXT='Select option':VM; CALL AVSS.MENU.FILL(AV,BASEAVMENUID,AVMENUID,HEAD.TEXT,PROMPT.LIST, ACTION.LIST,HELP.LIST); REPAINT='N'; CALL AVSS.MENU(AV,'1',BASEAVMENUID,CHOICE,REPAINT,'ON',SCREENPAINT); IF REPAINT#'N' THEN REPAINT.LINES<-1>=REPAINT; NOTE='Processing search...'; PRINT NOTEPRINT:NOTE; BEGIN CASE; CASE CHOICE='SEARCH'; GOSUB 400; * Compare the last search with current search if same end; C.LASTSEARCHITEM=C.SEARCHITEM; IF C.PREVIOUSSEARCHITEM<MATCHSESA.SEARCH.TEXT>=ITEM<MATCHSES A.SEARCH.TEXT> THEN; FOUNDLIST=C.ORIGFOUNDLIST; NUMFOUND=DCOUNT(FOUNDLIST,AM); GOSUB 100; * Filter "find" data in section

```
II and display window; C.SEARCHITEM=ITEM; END ELSE; FOUNDLIST=";
TOTHITLIST="; ID.STATUS='IDLIST.NOLIMIT';
SAVE.SEARCHSTRING=SEARCHSTRING; CALL
AVSS.FIND(AV,SEARCHSTRING,C.MATCHFINDFILEFN,C.MATCHFINDFILEDN,
C.MATCHFILEFN, C.FINDATTLIST,C.FIND.DEPTH,FOUNDLIST,ID.'
STATUS,'AVUP','N','ON'); IF ID.STATUS='EXIT' THEN GOTO 99;
C.ORIGFOUNDLIST=FOUNDLIST; NUMFOUND=DCOUNT(FOUNDLIST,AM); *
Loop through the ORIGINAL in savesearch; * and check searchstring for the value or
default to 0; NUMORIGWORDS=DCOUNT(SAVE.SEARCH STRING, ' ' ); FOR 1=1
TO NUMORIGWORDS; ORIGWORD=FIELD(SAVE.SEARCHSTRING,' ',I);
LOCATE(ORIGWORD,SEARCHSTRING,1;LOCVAL) THEN;
TOTHITLIST<1,-1>=SEARCHSTRING<1,LOCVAL>;
TOTHITLIST<2,-1>=SEARCHSTRING<2,LOCVAL>; END ELSE;
TOTHITLIST<1,-1>=ORIGWORD; TOTHITLIST<2,-1>=0; ENU; NEXT I; * Build
total information for window; NUMHITS=DCOUNT(TOTHITLI ST<1 >,VM); FOR
I=1 TO NUMHITS; TEXT<1,-1>=TOTHITLIST<2,I> 'R#5':'' occurences of
"':TOTHITLIST<1,1>:""'; NEXT I; TEXT<1,-1>=VM:NUMFOUND'R#5':"
matches on "':SAVE.SEARCHSTRING:""'; C.SEARCHTEXT=TEXT; GOSUB 100; *
Filter "find" data in section 11 and display window; C.SEARCHITEM=ITEM; END;
CASE CHOICE='RETURN'; GOSUB 200; * Clear and reset section 11;
ITEM=C.LASTSEARCHITEM; CALL
AVUP.FILL(AV,MATCHSESA.INVENTORY,",0," ); CALL
AVUP.FILL(AV,MATCHSESA.CARRIER,",0," ); CALL
AVUP.FILL(AV,MATCHSESA.COMMODITY,",0," ); CALL AVUP.CON(AV);
C.LASTSEARCHITEM="; END CASE; C.PREVIOUSSEARCHITEM=ITEM;
REPAINT.LINES<-1>=3:VM:24; 99*; NOTE=SPACE(39); PRINT
NOTEPRINT:NOTE; RETURN; 100* Filter "find" data base on criteria entered in section
I; FOR 1=NUMFOUND TO 1 STEP -1; MATCHFILEID=FOUNDLIST<I>; READ
MATCHFILEITEM FROM C.MATCHFILEFN,MATCHFILEID ELSE
MATCHFILEITEM="; BEGIN CASE; CASE
C.FILE.TO.USE='MATCHING.SHORTAGE';
CARRID=MATCHFILEITEM<MATCHSH.CARRIER>;
COMMID=MATCHFILEITEM<MATCHSH.COMMODITY.CODE>;
QUANTITY=MATCHFILEITEM<MATCHSH.QUANTITY>;
```

WEIGHT=MATCHFILEITEM<MATCHSH.WEIGHT>;
SHORTAGE.DATE=MATCHFILEITEM<MATCHSH.UPLOAD.DATE>;
PRO.NUMBER.DATE=MATCHFILEITEM<MATCHSH.UPLOAD.DATE>;
CLOSE.DATE=MATCHFILEITEM<MATCHSH.CLOSE.DATE>; CASE
C.FILE.TO.USE='MATCHING.INVENTORY';
CARRID=MATCHFILEITEM<MATCHINVEN.CARRIER>;
COMMID=MATCHFILEITEM<MATCHINVEN.COMMODITY.CODE>;
QUANTITY=MATCHFILEITEM<MATCHINVEN.QUANTITY>;
WEIGHT=MATCHFILEITEM<MATCHINVEN.WEIGHT>;
SHORTAGE.DATE=MATCHFILEITEM<MATCHINVEN.RECEIVE.DATE>; PRO.
NUMBER .DATE=MATCHFILEITEM<MATCHI NVEN .RECEIVE .DATE>;
CLOSE.DATE=MATCHFILEITEM<MATCHINVEN.CLOSE.DATE>; END CASE; *
Check for close date; IF CLOSE.DATE#" THEN;
FOUNDLIST=DELETE(FOUNDLIST,I); GOTO 189; END; * ChecK carrIer; IF
ITEM<MATCHSESA.CARRIER>#" THEN;
LOCATE(CARRID,ITEM,MATCHSESA.CARRIER;LOCVAL) ELSE;
FOUNDLIST=DELETE(FOUNDLIST,I); GOTO 189; END; END; * Check commodity
code; IF ITEM<MATCHSESA.COMMODITY>#" THEN;
LOCATE(COMMID,ITEM,MATCHSESA.COMMODITY;LOCVAL) ELSE;
FOUNDLIST=DELETE(FOUNDLIST,I); GOTO 189; END; END; * Check quant1ty; IF
ITEM<MATCHSESA.QUANTITY.TO>#" THEN; IF
QUANTITY<ITEM<MATCHSESA.QUANTITY.FROM> 0R
QUANTITY>ITEM<MATCHSESA.QUANTITY.TO> THEN;
FOUNDLIST=DELETE(FOUNDLIST,I); GOTO 189; END; END; * Check Weight; IF
ITEM<MATCHSESA.WEIGHT.FROM>#" THEN; IF
WEIGHT<ITEM<MATCHSESA.WEIGHT.FROM> 0R
WEIGHT>ITEM<MATCHSESA.WEIGHT.TO> THEN;
FOUNDLIST=DELETE(FOUNDLIST,I); GOTO 189; END; END; * Check Pro number
date; IF ITEM<MATCHSESA.PRO.NUMBER.DATE.FROM>#" THEN; IF
PRO.NUMBER.DATE<ITEM<MATCHSESA.PRO.NUMBER.DATE.FROM> OR
PRO.NUMBER.DATE>ITEM<MATCHSESA.PRO.NUMBER.DATE.TO> THEN;
FOUNDLIST=DELETE(FOUNDLIST,I); GOTO 189; END; END; * Check shortage
record created date; IF
ITEM<MATCHSESA.SHORTAGE.RECORD.CREATE.FROM>#" THEN; IF

SHORTAGE.DATE<ITEM<MATCHSESA.SHORTAGE.RECORD.CREATE.FROM>
OR
SHORTAGE.DATE>ITEM<MATCHSESA.SHORTAGE.RECORD.CREATE.TO>;
THEN; FOUNDLIST=DELETE(FOUNDLIST,I); GOTO 189; END; END; * Thus far, the MATCHINVENID has NOT been removed from the FOUNDLIST.; * If it had been removed, this section would be ommitted by the GOTO; * 189 code. Therefore, call the routine to build the special; * description field.; ***CALL AVSU.Z.MATCHSESA.DESCRIPTION (AV, COMMI D,MATCHFILEITEM, TEXT); ***ITEM<MATCHSESA.DESCRIPTION,I>=TEXT; 189*; NEXT I; GOSUB 200; * Clear and reset section 11; CONVERT AM TO VM IN FOUNDLIST;
ITEM<MATCHSESA.INVENTORY>=FOUNDLIST;
NUMIDS=DCOUNT(ITEM<MATCHSESA.INVENTORY>,VM); FOR 1=1 TO NUMIDS; MATCHFILEID=ITEM<MATCHSESA.INVENTORY,I>; READ MATCHFILEITEM FROM C.MATCHFILEFN,MATCHFILEID ELSE MATCHFILEITEM=''; BEGIN CASE; CASE C.FILE.TO.USE='MATCHING.INVENTORY';
ITEM<MATCHSESA.DESCRIPTION,I>=MATCHFILEITEM<MATCHINVEN.DESCRIPTION>;
ITEM<MATCHSESA.QUANTITY,I>=MATCHFILEITEM<MATCHINVEN.QUANTITY>;
ITEM<MATCHSESA.WEIGHT,I>=MATCHFILEITEM<MATCHINVEN.WEIGHT>;
ITEM<MATCHSESA.SHIPPER.INFORMATION,I>=MATCHFILEITEM<MATCHINVEN.SHIPPER.INFORMATION>;
ITEM<MATCHSESA.CONSIGNEE.INFORMATION,I>=MATCHFILEITEM<MATCHINVEN.CONSIGNEE.INFORMATION>; CASE C.FILE.TO.USE='MATCHING.SHORTAGE';
ITEM<MATCHSESA.DESCRIPTION,I>=MATCHFILEITEM<MATCHSH.DESCRIPTION>;
ITEM<MATCHSESA.QUANTITY,I>=MATCHFILEITEM<MATCHSH.QUANTITY>;
ITEM<MATCHSESA.WEIGHT,I>=MATCHFILEITEM<MATCHSH.WEIGHT>;
ITEM<MATCHSESA.SHIPPER.INFORMATION,I>=MATCHFILEITEM<MATCHSH.SHIPPER.INFORMATION>;

```
ITEM<MATCHSESA.CONSIGNEE.INFORMATION,I>=MATCHFILEITEM<MAT
CHSH.CONSIGNEE.INFORMATION>; END CASE; NEXT I; CALL
AVUP.FILL(AV,MATCHSESA.INVENTORY, " ,0, " ); CALL AVUP.CON(AV); *
Display window with total information; IF ITEM<MATCHSESA.CARRIER>#" OR
ITEM<MATCHSESA.COMMODITY>#" OR
ITEM<MATCHSESA.QUANTITY.FROM>#" OR
ITEM<MATCHSESA.WEIGHT.FROM>#" OR
ITEM<MATCHSESA.PRO.NUMBER.DATE.FROM>#" OR
ITEM<MATCHSESA.SHORTAGE.RECORD.CREATE.FROM>#" THEN;
CRITERIA.TEXT=',; CRITERIA.TEXT<1,-1>=VM:SPACE(6):'Limiting criteria
':VM; IF TEXT=" THEN TEXT=C.SEARCHTEXT;
NUMLIST=DCOUNT(ITEM<MATCHSESA.INVENTORY>,VM); IF
ITEM<MATCHSESA.CARRIER>#" THEN; LINER=SPACE(8):'Carrier"L#25':': ';
NUMCARRIERS=DCOUNT(ITEM<MATCHSESA.CARRIER>,VM); FOR 11=1 TO
NUMCARRIERS; IF 11#1 THEN LINER=LINER:', ';
LINER=LINER:ITEM<MATCHSESA.CARRIER,11>; NEXT 11;
CRITERIA.TEXT<1,-1=LINER; END; IF ITEM<MATCHSESA.COMMODITY>#"
THEN; LINER=SPACE(8):'Commodity"L#25':': ';
NUMCOMMODITIES=DCOUNT(ITEM<MATCHSESA.COMMODITY>,VM); FOR
11=1 TO NUMCOMMODITIES; IF 11#1 THEN LINER=LINER:', ';
LINER=LINER:ITEM<MATCHSESA.COMMODITY,11>; NEXT 11;
CRITERIA.TEXT<1,-1>=LINER; END; IF
ITEM<MATCHSESA.QUANTITY.FROM>#" THEN
CRITERIA.TEXT<1,-1>=SPACE(8):'Quantity" L#25':': ':'From
':OCONV(ITEM<MATCHSESA.QUANTITY.FROM,1>'MRO'),R#8l:, To
':OCONV(ITEM<MATCHSESA.QUANTITY.TO,1>,'MRO')'R#8'; IF
ITEM<MATCHSESA.WEIGHT.FROM>#" THEN
CRITERIA.TEXT<1,-1>=SPACE(8):'Weight''L#25':': ':'From
':OCONV(ITEM<MATCHSESA.WEIGHT.FROM,1>,'MRO')'R#8,:, To
':OCONV(lTEM<MATCHSESA.WEIGHT.TO'1>,'MRO')iR#8'; IF
ITEM<MATCHSESA.PRO.NUMBER.DATE.FROM>#'' THEN
CRITERIA.TEXT<1,-1>=SPACE(8):'Pro number date" L#25':': ,:,from '
:OCONV(ITEM<MATCHESA.PRO.NUMBER.DATE.FROM,1>,'DZ-')'R#8':'TO
OCONV(ITEM<MATCHSESA.PRO.NUMBER.DATE.TO,1>,'D2-')'R#8'; IF
```

ITEM<MATCHSESA.SHORTAGE.RECORD CREATE-FROM>#', THEN CRITERIA.TEXT<1,-1>=SPACE(8):'Shortage record create"L#25':': ':'From; ':OCONV(ITEM<MATCHSESA.SHORTAGE.RECORD.CREATE.FROM,1>'D2-')'R#8':' To ':OCONV(ITEM<MATCHSESA.SHORTAGE.RECORD.CREATE.TO,1,'D2-')'R#8'; CRITERIA.TEXT<1,-1>=VM:NUMLIST'R#5':' item(s) selected out of ':NUMFOUND:' matches based on search criteria'; TEXT<1,-1>=CRITERIA.TEXT; END; WIN='HELP.AW P'; CALL AVSS.WINDOW.TEXT(AV,WIN,TEXT,'Your search results :':VM,'N','ON'); RETURN; 200* Clear and reset section two; * Avgen manipulation !!!!!; * I am first subtracting the value position (LINE<1,2>) from the; * Screen position of the current value of section 2 (LNUM<1,2>) this; * will change to my LNUM value to its original default from the; * dictionary. I then hardcode the value position of LINE<1,2> to; * 1 SO that the cursor is on the first value every time a new search is; * performed. This is necessary in the event that a user moves the; * position of section 2 and the performs a new search, the system might; * build a shorter section 2 and leave the cursor stranded in section on; * a value with no data! (Ugly !); LNUM<1,2>=LNUM<1,2>-(LINE<1,2>-1); LINE<1,2>=1; NUMVAL.SECTION2=DCOUNT(ITEM<MATCHSESA.INVENTORT>, VM); CALL AVUP.COLUMN.VALUE.ADJUST(AV,2,'D',1,NUMVAL,SECTION2); CALL AVUP.FILL(AV,MATCHSESA.INVENTORY,",0,"); CALL AVUP.CON(AV); RETURN; 300* Create temp string to eliminate any special characters; * Create a temp string with only words in the string - eliminate all; * special find characters.; TEMPLEN=LEN(TEMPSIKING); CHR.POS=INDEX(TEMPSTRING,'-',1); TEMPSTRING=TEMPSTRING[1,CHR.POS-1]:TEMPSTRING[CHR.POS+1,TEMPLEN]; TEMPLEN=LEN(TEMPSTRING); LOOP; CHR.POS=INDEX(TEMPSTRING,'|',1); UNTIL CHR.POS=0 DO; TEMPSTRING=TEMPSTRING[1,CHR.POS-1]:' ':TEMPSTRING[CHR.POS+1,TEMPLEN]; REPEAT; TEMPSTRING=TRIM(TEMPSTRING); RETURN; 400* Compare the last search with current search if same end; IF C.PREVIOUSSEARCHITEM=ITEM THEN; DISPLAY.MESSAGE='Search criteria has not been changed'; RETURN TO 99; END; RETURN; AVSU.L.MATCHSE.1; SUBROUTINE AVSU.L.MATCHSE.1(AV); * Purpose :; * Status :5.81.A 07-01-97 14:17:53 174A6FD5; * Notes :A) 07-01-97 FWN move to live; * Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV AVSU.N.MATCHSE; INCLUDE AV.EOUATE MATCHING.SEARCH; INCLUDE

AVMP AVAL.EXEC.LEVEL; CALL
AVSU.Z.MATCHING.FIND(AV,MATCHSE.CARRIER,C.COMMODITY.CODE,C.MA
TCHINVENDN,C.MATCHINVENFN,'MATCHI
NG.INVENTORY',MATCHSE.MATCHED;
.INVENTORY,C.MATCHINVENFIDN,C.MATCHINVENFIFN); 99*; RETURN;
AVSU.L.MATCHSE.2; SUBROUTINE AVSU.L.MATCHSE.2(AV); * Purpose:; * Status
:6.00.A 07-01-97 14:19:51 176E0966; * Notes :A) 07-01-97 FWN move to live; *
Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV
AVSU.N.MATCHSE; I NCLUDE AV . EQUATE MATCHING . SEARCH; INCLUDE
AVMP AVAL.EXEC.LEVEL;
INVENID=ITEM<MATCHSE.MATCHED.INVENTORY,CURRVAL>; ;
WSID=AV<1,1,1>:'+':EXEC.LEVEL+1; WSITEM='LOOKUP'; CALL
AVSS.WORKSPACE(AV,WSID,WSITEM,'ON'); EXECTEXT='AV UP
':C.MATCHING.DICTIONARY:':MATCHING.INVENTORY VIEW ': INVENID;
CALL AVSS.EXECUTE(AV,EXECTEXT,'N', '', ' '); 99*; RETURN;
AVSU.S.MATCHSESA; SUBROUTINE AVSU.S.MATCHSESA(AV); * Purpose :Postsub
for matching; * Status :6.00.J 08-17-97 17:21:30 4F33E423; * Notes :A) 04-21-97 FWN
created>>:B) 04-22-97 RAR move to avsu.recov>>:C) 06-12-97 FWN created
synonym list>>:D) 06-16-97 FWN changed the methodology of reading from the
synonym file>>:E) 07-01-97 FWN create section that will limit carrier field to a specific
account>>:F) 07-02-97 FWN added variable to common variables & assigned ID to
it>>:G) 08-07-97 FWN added code for new search routine>>:H) 08-13-97 RAR add
problem.found trick from loadsub>>:l) 08-15-97 RAR continue>>:J) 08-17-97 RAR
continue; * Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV
AVSU.C.MATCHSESA; INCLUDE AV.EQUATE AV.DICT.STRUCTURE; INCLUDE
AV.EQUATE MATCHING.SEARCH.STANDALONE; INCLUDE AVMP
AVAL.EXEC.LEVEL; WSID=AV<1,1,1>:'+':EXEC.LEVEL; WSITEM=''; CALL
AVSS.WORKSPACE(AV,WSID,WSITEM,'OFF'); BEGIN CASE; CASE
WSITEM<1>#''; C.FILE.TO.USE=WSITEM<1>; IF
C.FILE.TO.USE#'MATCHING.SHORTAGE' AND
C.FILE.TO.USE#'MATCHING.INVENTORY' THEN; DISPLAY.MESSAGE='The
':C.FILE.TO.USE:' is an improper file for the matching search standalone process.,;
PROBLEM.FOUND=1; GOTO 99; END;
ITEM<MATCHSESA.SEARCH.TEXT>=WSITEM<2>;

```
ITEM<MATCHSESA.QUANTITY.FROM>=WSITEM<3>;
ITEM<MATCHSESA.QUANTITY.TO>=WSITEM<3>;
ITEM<MATCHSESA.WEIGHT>=WSITEM<4>;
ITEM<MATCHSESA.CARRIER>=WSITEM<5>; CASE WSITEM='';
AVMENUID='AVUP.CHOOsE.CENTER.sMALL.':AV<1,1,1>:'.':EXEC.LEVEL;
BASEAVMENUID='AVUP.CHOOSE.CENTER.SMALL.TEMPLATE.PORT';
HEAD.TEXT='Choose an option':VM; PROMPT.LIST='Search Overage
file':VM:'Search Shortage file';
ACTION.LIST='MATCHING.INVENTORY':VM:'MATCHING.SHORTAGE';
HELP.LIST=VM; CALL
AVSS.MENU.FILL(AV,BASEAVMENUID,AVMENUID,HEAD.TEXT,PROMPT.LIST,
ACTION.LIST,HELP.LIST); REPAINT='N'; CALL
AVSS.MENU(AV,'1',AVMENUID,C.FILE.TO.USE,REPAINT,'ON',SCREENPAINT);
IF REPAINT#'N' THEN REPAINT.LINES<-1>=REPAINT; IF
C.FILE.TO.USE='EXIT' THEN; PROBLEM.FOUND=1; GOTO 99; END; END
CASE; BEGIN CASE; CASE C.FILE.TO.USE='MATCHING.SHORTAGE'; OPEN
'MATCHING.SHORTAGE' TO C.MATCHFILEFN ELSE
CALLAVSS.ERROR(AV,'NOFILE','MATCHING.SHORTAGE'); OPEN
'DICT','MATCHING.SHORTAGE' TO MATCHFILEDN ELSE CALL
AVSS.ERROR(AV,'NOFILE','MATCHING.SHORTAGE'); OPEN
'MATCHING.SHORTAGE.FIND' TO C.MATCHFINDFILEFN ELSECALL
AVSS.ERROR(AV,'NO; FILE','MATCHING.SHORTAGE.FIND'); OPEN
'DICT','MATCHING.SHORTAGE.FIND' TO C.MATCHFINDFILEDN ELSE CALL
AVSS.ERROR; (AV,'NOFILE','MATCHING.SHORTAGE.FIND');
ITEM<MATCHSESA.FILE.LABEL>=' Shortage Search'; CASE
C.FILE.TO.USE='MATCHING.INVENTORY'; OPEN 'MATCHING.INVENTORY'
TO C.MATCHFILEFN ELSE CALL
AVSS.ERROR(AV,'NOFILE','MATCHING.INVENTORY'); OPEN
'DICT','MATCHING.INVENTORY' TO MATCHFILEDN ELSE CALL
AVSS.ERROR(AV,'NOFILE','MATCHING.INVENTORY'); OPEN
'MATCHING.INVENTORY.FIND' TO C.MATCHFINDFILEFN ELSE CALL
AVSS.ERROR(AV,'NOFILE';
,'MATCHING.INVENTORY.FINDI); OPEN 'DICT','MATCHING.INVENTORY.FIND' TO
C.MATCHFINDFILEDN ELSE CALL AVSS.ERR;
```

OR(AV,'NOFILE','MATCHING.INVENTORY.FIND');
ITEM<MATCHSESA.FILE.LABEL>=' Overage Search'; END CASE; * Get find depth from diet of matching.inventory.find; READY C.FIND.DEPTH FROM C.MATCHFINDFILEDN,'*STRUCTURE',AVDSTR.ID.LENGTH ELSE C.FIND.DEPTH=2; * Get attribute list to help build c.findattlist; READY TEMPATTLIST FROM MATCHFILEDN,'*ATTLIST',1 ELSE TEMPATTLIST=''; * Get find attribute list from diet of matching.inventory; READ STRUCTITEM FROM MATCHFILEDN,'*STRUCTURE' ELSE STRUCTITEM=',;
LOCATE(C.FILE.TO.USE:'.FIND',STRUCTITEM,AVDSTR.INDEX.FILE;LOCVAL) THEN;
INDEXATTLIST=STRUCTITEM<AVDSTR.INDEX.ATTRIBUTE>;
NUMINDEXATTS=DCOUNT(INDEXATTLIST,','); FOR 1=1 TO NUMINDEXATTS;
INDEXATT=FIELD(INDEXATTLIST,',',I); IF INDEXATT#'-ID' THEN;
LOCATE(INDEXATT,TEMPATTLIST,1;LOCVAL) IdCN;
C.FINDATTLIST<1,1,-1>=LOCVAL; END; END; NEXT I; END; *** USE TESTMATCH in AVXX to run this test!!!!!!; ***; OPEN 'MATCHING.SYNONYM' TO C.MATCHSYNFN ELSE CALL AVSS.ERROR(AV,'NOFILE','MATCHING.SYNONYM');
C.LASTSEARCHITEM=''; C.ORIGFOUNDLIST=''; C.PREVIOUSSEARCHITEM='';
ACCOUNT.NAME=AV<1,1,Z>; PORT=AV<1,1,1>; WSID=PORT:'+':EXEC.LEVEL;
BEGIN CASE; CASE ACCOUNT.NAME='CTS'; CALL
AVUP.ATTRIBUTE.MOulFY(AV,MATCHSESA.CARRIER,'REQUIRED.BLOCK','B');
ITEM<MATCHSESA.CARRIER,1>='15'; ITEM<MATCHSESA.CARRIER,2>='25';
ITEM<MATCHSESA.CARRIER,3>='30'; CASE ACCOUNT.NAME='NW'; CALL
AVUP.ATTRIBUTE.MODIFY(AV,MATCHSESA.CARRIER,'REQUIRED.BLOCK','B');
ITEM<MATCHSESA.CARRIER,1>='11'; CASE ACCOUNT.NAME='YFS'; CALL
AVUP.ATTRIBUTE.MODIFY(AV,MATCHSESA.CARRIER,'REQUIRED.BLOCK','B');
ITEM<MATCHsEsA.cARRIER,1>='02'; CASE 1; CALL
AVUP.ATTRIBUTE.MODIFY(AV,MATCHSESA.CARRIER,'REQUIRED.BLOCK',"); END CASE; 99*; RETURN; AVSU . P. COM. LWC; SUBROUTINE AVSU.P.COM.CODE(AV); * Purpose :Postsub - Matching Commodity; Status :5.81.A 07-01-97 16:59:48 99821F5; * Notes :A) 07-01-97 RAR move to live & recompile; * Argument :1) AV [P] Avexxis system info; INCLUDE AVIC AVIC.AVUP; INCLUDE AV.EQUATE COMMODITY; INCLUDE AV.EQUATE MATCHING.COMMODITY; INCLUDE AVMP AVAL.EXEC.LEVEL; IF NOT(TEXTCHG) THEN RETURN; READ MCOMMITEM FROM REFFN(COM.CODE),CURRTEXT ELSE RETURN;

DICT.TO.LOAD=MCOMMITEM<MCOMM.INVENTORY.DICT>; * Populate the workspace with the commod~ty cooe; WSID=AV<1,1,1>:'+':EXEC.LEVEL+1; WSITEM=CURRTEXT:AM:DICT.TO.LOAD; CALL AVSS.WORKSPACE(AV,WSID,WSITEM,'ON'); CALL AVSS.EXECUTE(AV,'AV UP ':DICT.TO.LOAD:':MATCHING.INVENTORY NEW','N', '', ''); CALL AVUP.CLEAR(AV,CURRATT,CURRVAL,'N'); REPAINT.LINES<1,-1>=1:VM:24; 99*; RETURN; AVSU.P.COMSRCH.CODE; SUBROUTINE AVSU.P.COMSRCH.CODE(AV); * Purpose :Postsub - Matching Commodity Search; * Status :5.81.A 07-01-97 17:01:13 99D023A; * Notes :A) 07-01-97 RAR move to live & recompile; * Argument :1) AV [P] Avexxis system info; INCLUDE AVIC AVIC.AVUP; INCLUDE AV.EQUATE COMMODITY.SEARCH; INCLUDE AV.EQUATE MATCHING.COMMODITY; INCLUDE AVMP AVAL.EXEC.LEVEL; IF NOT(TEXTCHG) THEN RETURN; READ MCOMMITEM FROM REFFN(COMSRCH.CwE),CURRTEXT ELSE RETURN; DICT.TO.LOAD=MCOMMITEM<MCOMM.SEARCH.FILE>; WSID=AV<1,1,1>:'+':EXEC.LEVEL+1; WSITEM=CURRTEXT:AM:MCOMMITEM<MCOMM.INVENTORY.DICT>; CALL AVSS.WORKSPACE(AV,WSID,WSITEM,'ON'); CALL AVSS.EXECUTE(AV,'AV UP ':DICT.TO.LOAD:':MATCHING.SEARCH NEW','N', '', ''); CALL AVUP.CLEAR(AV,CURRATT,CURRVAL, 'N'); REPAINT.LINES<1,-1>=1:VM:24; 99*; RETURN; AVSU. B. MATCHINVEN; SUBROUTINE AVSU.B.MATCHINVEN(AV); * Purpose :Backend of Matching. inventory - locK PW recorci, update PGU record unlock PGU record, update inventory record, ar; Unlock inventory record; print labels.; * Status :6.00.F 04-13-98 06:39:45 C1EA3D3B; * Notes :A) 07-03-97 RAR initial version>>:B) 08-02-97 FWN added code to unlock inventory record>>:C) 08-17-97 RAR move to I; >>:D) 02-26-98 NDL Add datrash log code>>:E) 04-01-98 NDL Put mult.carr and datrash into prod>>:F) 04-13-98 NDL aW account to trash; Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV AVSU.N.MATCHINVEN; INCLUDE AV.EQUATE AV.PORT; INCLUDE AV.EQUATE INVENTORY; INCLUDE AV.EQUATE MATCHING. INVENTORY; INCLUDE AV.EQUATE PORTABLE.GROUP; INCLUDE AVMP AVAL.EXEC.LEVEL; I NVEN ID= I TEM<MATCH I NVEN. I NVENTORY>; PORTABLE. GROUP.ADDED=0; PORTABLE. GROUP.DELETED=0; PORT=AV<1,1,1>; PASS.PORTEXEC=PORT:'+':EXEC.LEVEL; CALL AVSS.FILE.NAME(AV,'INVENTORY', INVENTORY.ACCOUNT, ''); CALL AVSS. FILE.NAME(AV,'PORTABLE.GROUP',PORTABLE.GROUP.ACCOUNT, ''); * Update PGU

records and inventory recorcl Wlth new PGU info.; IF C.ORIGPGULIST#C.PGULIST AND INVENID#" THEN; * Loop through the original list of portable groups.; NUMORIGPGULIST=DCOUNT(C.ORIGPGULIST<1>,VM); FOR I = 1 TO NUMORIGPGVLIST; PGUID=C.ORIGPGULIST<1,1>; I F PGUID#" THEN; LOCATE(PGUID,C.PGULIST,1;TEMPVAL) THEN; QTY.CHANGE=C. PGUL IST<2, TEMPVAL>-C.OR! GPGUL I ST<2,1 >; GOSUB 100; * Lock the Portable group record.; GOSUB 200; * Update the Portable group record.; GOSUB 300; * Unlock the Portable group record.; C.PGULIST<4,TEMPVAL>='X'; END ELSE; PORTABLE . GROUP .DELETED=1; GOSUB 100; * Lock the Portable group record.; GOSUB 200; * Update the Portable group record.; GOSUB 300; * Unlock the Portable group record.; END; END; PORTABLE . GROUP. DELETED=0; NEXT I; * Loop through the updated l ist of portable groups.; NUMPGUL I ST=DCOUNT(C.PGULIST<1> ,VM); FOR 1=1 TO NUMPGULIST; PGUID=C.PGULIST<1, 1>; IF PGUID#" AND C.PGULIST<4,1>='' THEN; PORTABLE.GROUP.ADDED=1; QTY.CHANGE=C.PGULIST<2,1>; GOSUB 100; * Lock the Portable group record.; GOSUB 200; * Update the Portable group record.; GOSUB 300; * Unlock the Portable group record.; END; PORTABLE.GROUP.ADDED=0; NEXT I; * Update inventory record.; C.INVENITEM<INVEN.PORTABLE.GROUP>=C.PGULIST<1>; C.INVENITEM<INVEN.QTY.ON.HAND>=C.PGULIST<2>; C.INVENITEM<INVEN.QTY.COMMITTED>=C.PGULIST<3>; WRITE C.INVENITEM ON C.INVENFN,INVENID; END; * Make sure that the inventory record is unlocked!!!; IF INVENID#" THEN; CALL AVSS.LOCK(AV,AVLOCKFN,INVENTORY.ACCOUNT,'INVENTORY',INVENID,'OFF',PASS.PORTEXEC); END; ***** * ********** *******
**********; * Print 1nventory lanels; NUMBER.LABELS=ITEM<MATCHINVEN.QUANTITY>-ORIGITEM<MATCHINVEN.QUANTITY>; IF NUMBER.LABELS>0 THEN; CALL AVSS.AV.PORT(AV,AVPORTITEM); IDATE=DATE(); EDATE=OCONV(IDATE,'D2-'); TYPE=AVPORTITEM<AVPORT.PRINTER.DEVICE; * The inventory label printing program is expecting to print labels; * for an inventory record only !!!. Therefore, we must "fool'' the; * system by stuffing the description line to print into a dummy; * inventory record in the right field.; INVENIIEM='';
INVENITEM<INVEN.DESCRIPTION>=ITEM<MATCHINVEN.PRODUCT>; IF TYPE='TEC' OR TYPE='BB.SCOS2' THEN; PRICE=''; CALL AVSU.Z.INVEN.LABEL.PRINT(AV,ID,INVENITEM,NUMBER.LABELS,EDATE,PRICE,TYPE); END; END; 99*; RETURN; 100* Lock the Portable group record.; IF PGUID#" THEN;

LOCK.COUNTER=0; LOOP; LOCKFLAG='ON'; CALL
AVSS.LOCK(AV,AVLOCKFN,PORTABLE.GROUP.ACCOUNT,'PORTABLE.GROUP',PGUI
D,LOCKFLAG,PASS.PORTEXEC); UNTIL LOCKFLAG#'LOCKED' DO;
LOCK.COUNTER=LOCK.COUNTER+1; PRINT BELL;; SLEEP 1; IF LOCK.COUNTER>5
THEN; DISPLAY.TEXT='The portable group record ':PGUID:' is locked. The process cannot
continue until the record is unlocked!; CALL
AVSS.DISPLAY.MESSAGE(AV,DISPLAY.TEXT,"); LOCK.COUNTER=0; END; REPEAT;
END; RETURN; 200* Update the Portable group record.; READ PGUITEM FROM
C.PGUFN,PGUID ELSE PGUITEM="; IF PORTABLE.GROUP.ADDED THEN;
NUMLINES=DCOUNT(PGUITEM<PGU.INVENTORY>,VM);
INSERT.LINE=NUMLINES+1; CALL
AVSS.COLUMN.VALUE.ADJUST(AV,'PORTABLE.GROUP',PGUITEM,2,'1',1NSERT.LIN
E,1); PGUITEM<PGU.INVENTORY,INSERT.LINE>=INVENID;
PGUITEM<PGU.QUANTITY,INSERT.LINE>=QTY.CHANGE; IF PGUID="DATRASH"
THEN; CALL AVSU.Z.ALL.PGU.DATRASH.LOG(AV,INVENID,QTY.CHANGE,"); END;
WRITE PGUITEM ON C.PGUFN,PGUID; END ELSE;
LOCATE(INVENID,PGUITEM,PGU.INVENTORY;LOCVAL) THEN; IF
PORTABLE.GROUP.DELETED THEN;
INVENID=PGUITEM<PGU.INVENTORY,LOCVAL>;
QTY.CHANGE=PGUITEM<PGU.QUANTITY,LOCVAL>;
PGUITEM=DELETE(PGUITEM,PGU.INVENTORY,LOCVAL);
PGUITEM=DELETE(PGUITEM,PGU.QUANTITY,LOCVAL); IF PGUID="DATRASH"
THEN; CALL AVSU.Z.ALL.PGU.DATRASH.LOG(AV,INVENID,QTY.CHANGE,"); END;
END ELSE; ORIGPGU.QTY=PGUITEM<PGU.QUANTITY,LOCVAL>;
PGUITEM<PGU.QUANTITY,LOCVAL>=ORIGPGU.QTY+QTY.CHANGE; IF
ORIGPGU.QTY+0=0 THEN; INVENID=PGUITEM<PGU.INVENTORY,LOCVAL>;
CHANGE=PGUITEM<PGU.QUANTITY,LOCVAL>; IF PGUID="DATRASH" THEN;
CALL AVSU.Z.ALL.PGU.DATRASH.LOG(AV,INVENID,CHANGE,"); END;
PGUITEM=DELETE(PGUITEM,PGU.INVENTORY,LOCVAL);
PGUITEM=DELETE(PGUITEM,PGU.QUANTITY,LOCVAL); END; END; WRITE
PGUITEM ON C.PGUFN,PGUID; END; END; RETURN; 300* Unlock the Portable group
record.; CALL
AVSS.LOCK(AV,AVLOCKFN,PORTABLE.GROUP.ACCOUNT,'PORTABLE.GROUP',PGUI
D,'OFF',PASS.PORTEXEC); RETURN; READ PGUITEM FROM C.PGUFN,PGUID ELSE

PGUITEM=''; LOCATE(INVENID,PGUITEM,PGU.INVENTORY;LOCVAL) THEN;
ORIGPGU.QTY=PGUITEM<PGU.QUANTITY,LOCVAL>;
PGUITEM<PGU.QUANTITY,LOCVAL>=ORIGPGU.QTY+QTY.CHANGE; IF
ORIGPGU.QTY+0=0 THEN;
PGUITEM=DELETE(PGUITEM,PGU.INVENTORY,LOCVAL);
PGUITEM=DELETE(PGUITEM,PGU.QUANTITY,LOCVAL); END; WRITE PGUITEM ON
C.PGUFN,PGUID; END; AVSU.P.MATCHINVEN.PORTABLE.GROUP; SUBROUTINE
AVSU.P.MATCHINVEN.PORTABLE.GROUP(AV); * Purpose: Check for committed quantity
before deleting a line; if quantity committed then do not alter line.; Status :6.00.C 08-17-97
20:45:57 2C82D019; * Notes :A) 08-02-97 FWN initial version>>:B) 08-02-97 FWN added
code to verity that the carrier Id is valid for the location,>>:C) 08-17-97 RAR continue; *
Argument :1) AV [P] Avexxis system info; INCLUDE AVIC AVIC.AVUP; INCLUDE
AV.EQUATE MATCHING. INVENTORY; INCLUDE AV.EQUATE PORTABLE.GROUP; IF
NOT(TEXTCHG) AND CHOICE#'DELV' THEN RETURN;
QTY.COMMITTED=ITEM<MATCHINVEN.PORTABLE.GROUP.QTY.COMMITTED,CU
RRVAL>; IF QTY.COMMITTED+0>0 THEN; PROBLEM.FOUND=1;
DISPLAY.MESSAGE='Quantity has been committed - The portable group cannot be changed
and the line cannot be deleted'; IF CHOICE#'DELV' THEN CALL
AVUP.CLEAR(AV,CURRATT,CURRVAL,'Y'); GOTO 99; END; IF CURRTEXT='' THEN;
PROBLEM.FoUND=1; DISPLAY.MESSAGE='Delete the current line and use insert to add
new portable group'; CALL AVUP.CLEAR(AV,CURRATT,CURRVAL,'Y'); GOTO 99; END;
PGUID=CURRTEXT; READ PGUITEM FROM
REFFN(MATCHINVEN.PORTABLE.GROUP),PGUID THEN;
PGUCARR=PGUITEM<PGU.CARRIER>; IF PGUCARR#'' THEN; IF
PGUCARR#ITEM<MATCHINVEN.CARRIER> THEN; DISPLAY.MESSAGE='Portable
group ':PGUID:' is for carrier ':PGUITEM<PGU.CARRIER>:'; this item (which is for carrier
':1TEM<MATCHINVEN.CARRIER>:') cannot be put into this Portable Group.'; CALL
AVUP.CLEAR(AV,CURRATT,CURRVAL,'Y'); PROBLEM.FOUND=1; RETURN; END;
END; END; 99*; RETURN; AVSU.P .MATCH INVEN . PORTABLE . GROUP .QOH;
SUBROUTINE AVSU.P.MATCHINVEN.PORTABLE.GROUP.QOH(AV); * Purpose: Verify
that the amount decreased is not less than the quantity committed, per line.; * Status :6.00.A
08-01-97 14:05:27 2548AEC5; * Notes :A) 08-01-97 FWN initial version; * Argument :1) AV
[P] Avexxis system info; INCLUDE AVIC AVIC.AVUP; INCLUDE AV.EQUATE
MATCHING. INVENTORY; IF NOT(TEXTCHG) AND CHOICE#'DELV' THEN RETURN;

IF CURRTEXT<0 AND CURRTEXT#" THEN; PROBLEM.FOUND=1; DISPLAY.MESSAGE='Negative quantities are not valid'; CALL AVUP.CLEAR(AV,CURRATT,CURRVAL,'Y'); GOTO 99; END; QTY.COMMITTED=ITEM<MATCHINVEN.PORTABLE.GROUP.QTY.COMMITTED,CURRVAL>; IF QTY.COMMITTED#" AND CURRTEXT#" THEN; IF CURRTEXT<QTY.COMMITTED THEN; PROBLEM.FOUND=1; DISPLAY.MESSAGE='The portable group quantity cannot be reduced below the quantity committed..; CALL AVUP.CLEAR(AV,CURRATT,CURRVAL,'Y'); END; END; IF CHOICE='DELV' THEN; IF QTY.COMMITTED+0>0 THEN; PROBLEM.FOUND=1; DISPLAY.MESSAGE='Quantity has been committed - the line cannot be deleted''; GOTO 99; END; END; IF CURRTEXT='' THEN; IF QTY.COMMITTED+0>0 THEN; PROBLEM.FOUND=1; DISPLAY.MESSAGE='Quantity has been committed - The portable group quantity cannot be reduced below the committed quantity; CALL AVUP.CLEAR(AV,CURRATT,CURRVAL,'Y'); GOTO 99; END; END; 99*; RETURN; AVSU.P.MATCHINVEN.3; SUBROUTINE AVSU.P.MATCHINVEN.3(AV); * Purpose: Check for committed quantity before deleting a line; if quantity committeci then cion~t aelete line.; * Status :6.00.A 08-01-97 16:26:31 24AE7480; * Notes :A) 08-01-97 FWN initial version; * Argument :1) AV [P] Avexxis system info; INCLUDE AVIC AVIC.AVUP; INCLUDE AV.EQUATE MATCHING. INVENTORY; IF NOT(TEXTCHG) AND CHOICE#'DELV' THEN RETURN; * PGUID=ITEM<MATCHINVEN.PORTABLE.GROUP,CURRVAL>; QTY.COMMITTED=ITEM<MATCHINVEN.PORTABLE.GROUP.QTY.COMMITTED,CURRVAL>; BEGIN CASE; CASE CURRATT=MATCHINVEN.PORTABLE.GROUP; IF QTY.COMMITTED+0>0 THEN; PROBLEM.FOUND=1; DISPLAY.MESSAGE='Quantity has been committed - The portable group cannot be changed and the line cannot be deleted'; IF CHOICE#'DELV' THEN CALL AVUP.CLEAR(AV,CURRATT,CURRVAL,'Y'); GOTO 99; END; IF CURRTEXT=" THEN; PROBLEM.FOUND=1; DISPLAY.MESSAGE='Delete the current line and use insert to add new portable group'; CALL AVUP.CLEAR(AV,CURRATT,CURRVAL,'Y'); GOTO 99; END; CASE CURRATT=MATCHINVEN.PORTABLE.GROUP.QTY.COMMITTED; PROBLEM.FOUND=1; DISPLAY.MESSAGE='The quantity committed cannot be changed'; CALL AVUP.CLEAR(AV,CURRATT,CURRVAL,'Y'); GOTO 99; END CASE; 99*; RETURN; AVSU P.MATCHINVEN.RELEASE; SUBROUTINE AVSU.P.MATCHINVEN.RELEASE(AV); * Purpose :To populate the release.by and release.date fields.; * Status :6.00.B 07-31-97 17:31:47 2683768E; * Notes :A) 07-30-97 FWN

initial version> >:B) 07-31-97 RAR add rowprint and remove repaint.lines; * Argument :1) AV [P] Avexxis system info; INCLUDE AVIC AVIC.AVUH; ; INCLUDE AV.EQUATE AV.USER; INCLUDE AV.EQUATE MATCHING. INVENTORY; IF NOT(TEXTCHG) AND CHOICE#'DELV' THEN RETURN; IF CURRTEXT='Y' THEN; OPEN 'AV.USER' TO AVUSRFN ELSE CALL AVSS.ERROR(AV,'NOFILE','AV.USER'); USER.ACCWNT=AV<1,1,2>; READV PERSONNEL .NUMBER FROM AVUSRFN ,USER .ACCOUNT,AVUSR.PERSONNEL .REFERENCE ELSE PERSONNEL . NUMBER="; ITEM<MATCHINVEN.RELEASE.BY> =PERSONNEL.NUMBER; ITEM<MATCHINVEN.RELEASE.DATE> =DATE(); ROWPRINT=1; END; IF CURRTEXT=" OR CHOICE='DELV' THEN; ITEM<MATCHINVEN . RELEASE . BY> = "; ITEM<MATCHINVEN.RELEASE.DATE> ="; ROWPRINT=1; END; 99*; RETURN; AVSU P.MATCHINVEN.MATCHED; SUBROUTINE AVSU.P.MATCHINVEN.MATCHED(AV); * Purpose :To populate the matched.by and matched.date fields.; * Status :6.00.B 07-31-97 17:30:38 2682FDDE; * Notes :A) 07-30-97 FWN initial version> >:B) 07-31-97 RAR remove repaint.lines and add rowprint; * Argument :1) AV [P] Avexxis system info; INCLUDE AVIC AVIC.AVUP; INCLUDE AV.EQUATE AV.USER; INCLUDE AV.EQUATE MATCHING. INVENTORY; IF NOT(TEXTCHG) AND CHOICE#'DELV' THEN RETURN; IF CURRTEXT='Y' THEN; OPEN 'AV.USER' TO AWSRFN ELSE CALL AVSS.ERROR(AV,'NOFILE','AV.USER'); USER.ACCOUNT=AV<1,1,2>; READV PERSONNEL .NUMBER FROM AWSRFN,USER .ACCOUNT ,AWSR.PERSONNEL .REFERENCE ELSE PERSONNEL .NUMBER="; ITEM<MATCHINVEN.MATCHED.BY> =PERSONNEL.NUMBER; ITEM<MATCHINVEN.MATCHED.DATE> =DATE(); ROWPRINT=1; END; IF CURRTEXT=" OR CHOICE='DELV' THEN; ITEM<MATCHINVEN.MATCHED.BY> = "; ITEM<MATCHINVEN.MATCHED.DATE> ="; ROWPRINT=1; END; 99*; RETURN; AVSU,P.MATCHINvtN.HuLu; SUBROUTINE AVSU.P.MATCHINVEN.HOLD(AV); * Purpose :To populate the hold.by and hold.date fields.; * Status :6.00.B 07-31-97 17:27:12 26725154; * Notes :A) 07-30-97 FWN initial version> >:B) 07-31-97 RAR remove repaint.line and use rowprint; * Argument :1) AV [P] Avexxis system info; INCLUDE AVIC AVIC.AVUP; INCLUDE AV.EQUATE AV.USER; INCLUDE AV.EQUATE MATCHING. INVENTORY; IF NOT(TEXTCHG) AND CHOICE#'DELV' THEN RETURN; IF CURRTEXT='Y' THEN; OPEN 'AV.USER' TO AVUSRFN ELSE CALL AVSS.ERROR(AV,'NOFILE','AV.USER'); USER.ACCOUNT=AV<1,1,2>; READY PERSONNEL.NUMBER FROM AWSRFN,USER.ACCOUNT,AVUSR.PERSONNEL.REFERENCE ELSE

PERSONNEL.NUMBER="; ITEM<MATCHINVEN.HOLD.BY> =PERSONNEL.NUMSER; ITEM<MATCHINVEN.HOLD.DATE> =DATE(); ROWPRINT=1; END; IF CURRTEXT=" OR CHOICE='DELV, THEN; ITEM<MATCHINVEN.HOLD.BY> = "; ITEM<MATCHINVEN.HOLD.DATE> ="; ROWPRINT=1; END; 99*; RETURN; AVSU_P.MATCHINVEN.INVENTORY; SUBROUTINE AVSU.P.MATCHINVEN.INVENTORY(AV); * Purpose :To retrieve portable group info from the inventory record.; * Status :6.00.H 08-17-97 16:42:23 70F24C9F; * Notes :A) 07-23-97 FWN initial version> >:B) 07-24-97 FWN corrections> >:C) 07-28-97 FWN create locking routine to lock inventory record> >:D) 08-02-97 FWN added section to build ORIGPGULIST> >:E) 08-02-97 FWN corrected the qty obtained from the inventory file> >:F) 08-07-97 RAR Rearrange include file> >:G) 08-15-97 RAR continue> >:H) 08-17-97 RAR UPDATE c.inventitem and correct usage of qty.initial; * Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV AVSU.N.MATCHINVEN; INCLUDE AV.EQUATE INVENTORY; INCLUDE AV.EQUATE MATCHING. INVENTORY; INCLUDE AVMP AVAL.EXEC.LEVEL; IF NOT(TEXTCHG) AND CHOICE#'DELV' THEN RETURN; OPEN 'MATCHING.INVENTORY.INDEX.INVENTORY' TO MATCHINVENXINVENFN ELSE CALL AVSS.ERROR(AV,'NOFILE','MATCHING.INVENTORY.INDEX.INVENTORY'); INVENID=CURRENT; ORIGPGULIST= "; READ MATCHINVENXINVENITEM FROM MATCHINVENXINVENFN,INVENID THEN; DISPLAY.MESSAGE='The inventory record ':INVENID:' has been assigned to another matching inventory record and it cannot be assigned!'; PROBLEM.FOUND=1; CALL AVUP.CLEAR(AV,CURRATT,CURRVAL,"); GOTO 99; END; MATCHINVENXINVENITEM=MATCHINVENXINVENITEM; * For the Avgen compiter.; IF INVENID#" THEN; LOCKFLAG='ON'; PORT=AV<1,1,1>; PASS.PORTEXEC=PORT:'+':EXEC.LEVEL; CALL AVSS.FILE.NAME(AV,'INVENTORY',INVENTORY.ACCOUNT,"); CALL AVSS.LOCK(AV,AVLOCKFN,INVENTORY.ACCOUNT,; 'INVENTORY',INVENID,LOCKFLAG,PASS.PORTEXEC); IF LOCKFLAG='LOCKED' THEN; DISPLAY.MESSAGE='The inventory record ':INVENID:' was locked, portable group information was not obtained'; PROBLEM.FOUND=1; CALL AVUP.CLEAR(AV,CURRATT,CURRVAL,"); GOTO 99; END; END; READ C.INVENITEM FROM C.INVENFN,INVENID ELSE C.INVENITEM=''; BEGIN CASE; ASK C.INVENITEM#" AND INVENID#" AND CHOICE#'DELV'; IF ITEM<MATCHINVEN.PORTABLE.GROUP.QOH>#" THEN GOSUB 100; ITEM<MATCHINVEN.PORTABLE.GROUP> =C.INVENITEM<INVEN.PORTABLE.GRO

UP>; CALL AVUP.FILL(AV,MATCHINVEN.PORTABLE.GROUP," ",","); ITEM<MATCHINVEN.PORTABLE.GROUP.QOH>=C.INVENITEM<INVEN.QTY.ON.HAND>; CALL AVUP.FILL(AV,MATCHINVEN.PORTABLE.GROUP.QOH,",","); ITEM<MATCHINVEN.PORTABLE.GROUP.QTY.COMMITTED>=C.INVENITEM<INVEN.QTY.COMMITTED>; CALL AVUP.FILL(AV,MATCHINVEN.PORTABLE.GROUP.QTY.COMMITTED,",","); * Build ORIGPGULIST with the origianl inventory amounts then pass; * variable in common to be used in the wrapup to verify that the; * total PGU quantity has not been changed.; ORIGPGULIST<1>=ITEM<MATCHINvEN. PUR I ABLt. GKUUP>; ORIGPGULIST<2>=ITEM<MATCHINVEN.PORTABLE.GROUP.QOH>; ORIGPGULIST<3>=ITEM<MATCHINVEN.PORTABLE.GROUP.QTY.COMMITTED>; C.ORIGPGULIST=ORIGPGULIST; CASE INVENID=" OR CHOICE='DELV'; GOSUB 100;* Eliminate the Portable.group section.; END CASE; REPAINT.LINES<-1>=TM<1,2,1>:VM:BM<1,2,1>; CALL AVUP.REPAINT(AV); 99*; RETURN; 100* Eliminate the Portable.group section.; NUMLINES=DCOUNT(ITEM<MATCHINVEN.PORTABLE.GROUP>,VM); CALL AVUP.COLUMN.VALUE.ADJUST(AV,3,'D',1,NUMLINES); RETURN; AVSU.Z.MATCHSESA.DESCRIPTION; SUBROUTINE AVSU.Z.MATCHSESA.DESCRIPTION(AV'COMMODITY.CODE`MATCHINVENITEM,TEXT); * Purpose :Postsub for the creation of the concatinated description field.; * Status :6.00.A 07-31-97 13:47:46 28DA6ABF; * Notes :A) 07-31-97 FWN initial version; * Argument :1) AV [P] Avexxis system info>>:2) COMMODITY.CODE [P] The commodity.code from the matching.inventory file - used to select the fields to build the text description>>:3); MATCHINVENITEM [P] The record of the Matching~inventory file>>:4) TEXT [R] outine will pass the text description; to the calling routine; INCLUDE AVIC AVIC.AVUP; INCLUDE AV.EQUATE MATCHING. INVENTORY; TEXT=''; BEGIN CASE; CASE COMMODITY.CODE='0001'; TEXT=MATCHINVENITEM<MATCHINVEN.PRODUCT>; TEXT=TEXT:' ':MATCHINVENITEM<MATCHINVEN.MANUFACTURER>; TEXT=TEXT:' ':MATCHINVENITEM<MATCHINVEN.STYLE>; TEXT=TEXT:' ':MATCHINVENITEM<MATCHINVEN.MODEL>; TEXT=TEXT:' ':MATCHINVENITEM<MATCHINVEN.RN.NUMBER>; TEXT=TEXT:' ':MATCHINVENITEM<MATCHINVEN.SIZE>; CASE COMMODITY.CODE='0002'; TEXT=MATCHINVENITEM<MATCHINVEN.PRODUCT>; TEXT=TEXT:'

':MATCHINVENITEM<MATCHINVEN.MANUFACTURER>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.SIZE>; CASE COMMODITY.CODE='0003';
TEXT=MATCHINVENITEM<MATCHINVEN.PRODUCT>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.MANUFACTURER>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.SIZE>; CASE COMMODITY.CODE='0004';
TEXT=MATCHINVENITEM<MATCHINVEN.PRODUCT>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.MANUFACTURER>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.PATTERN>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.MATERIAL>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.WEAVE>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.SIZE>; CASE COMMODITY.CODE='0005';
TEXT=MATCHINVENITEM<MATCHINVEN.PRODUCT>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.MANUFACTURER>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.STYLE>; TEXT=TEXT:'
':MATCHINVENITEM<MATCHINVEN.MODEL>; END CASE; TEXT=TRIM(TEXT);
99*; RETURN; AVPR MATCHING.SHORTAGE.CONWAY.UPLOAD; * Purpose :Upload matching shortage information from diskette for the carrier conway; * Status :5.80.R 07-31-97 18:06:20 D65DOA83; * Notes :A) 02-22-96 RAR add termite transfer ability>>:B) 04-16-97 RAR initial version>>:C) 04-17-97 BAR comment out origin>>:)) 04-18-97 BAR comment out item.update>>:E) 04-18-97; RAR modify disk format for new diskette>>:F) 07-07-97 RAR initial version>>:; G) 07-08-97 FWN convert program to run for the Matching.shortage file>>:H) 07-09-97 FWN minor changes based on format changes to the file being uploaded>>:!); 07-10-97 FWN changed the conversion of numerical values (quantities)~:J) 07-11-97 FWN correction to Item.update; and new modifications per Bret>>:K) 07-15-97 FWN rebuilding the method for creating records due to carrier ; information changes>>:L) 07-16-97 FWN continued madification to based on changes made to file format by the carrier>>:M) 07-17-97 FWN r~; moved the attribute PRO.NUMBER.LINE from code b/c it was removed from dict>>:N) 07-18-97 FWN changes the carrier requested, revision date was added to track the date of any modifications to the record>>:O); 07-21-97 RAR remove use of pro.number.line index and replace with reading in records from; pro.number index and comparing each line>>:P) 07-21-97 RAR rename file prefixes>>:Q) 07-; 0-97 FWN recompile for change in field positions>>:R) 07-31-97 RAR add control account check; INPUT AV; INCLUDE AVIC AVIC.SETUP; INCLUDE AV.EQUATE MATCHING.SHORIAGt; INCLUDE AV.EQUATE MATCHING.UPLOAD.CONVERSION; INCLUDE AVMP

```
AVAL.EXEC.LEVEL; BADLIST=''; PRONUMLIST=''; CURRNEXTITEM='';
DATE.STAMP=DATE(); EMULATION=AV<1,1,3>; ENDCMD=CHAR(0);
ERROR.TEXT=''; LABEL.TEXT='Upload Carrier matching information from diskette';
IDLENGTH='7'; NOTE.PRINT=0; PCCMD=CHAR(27):CHAR(B); PORT=AV<1,1,1>;
PROGRESS.WINID='PROGRESS'; PROGRESS.STATUS='OPEN'; PROLIST=''; TOT=0;
OPEN 'MATCHING.SHORTAGE' TO MATCHSHFN ELSE CALL
AVSS.ERROR(AV,'NOFILE','MATCHING.SHORTAGE'); OPEN
'DICT','MATCHING.SHORTAGE' TO MATCHSHDN ELSE CALL
AVSS.ERROR(AV,'NOFILE','D_; MATCHING.SHORTAGE'); OPEN
'MATCHING.SHORTAGE.UPLOAD' TO MATCHSHUFN ELSE CALL
AVSS.ERROR(AV,'NOFILE','MATCHING.SHORTAGE.UPLOAD'); OPEN
'MATCHING.SHORTAGE.INDEX.PRO.NUMBER' TO MATCHSHXPNFN ELSE CALL
AVSS.ERROR(AV,'NOFILE','MATCHING.SHORTAGE.INDEX.PRO.NUMBER); OPEN
'MATCHING.UPLOAD.CONVERSION' TO MATCHUPCONFN ELSE CALL
AVSS.ERROR(AV,'NOFILE','; MATCHING.UPLOAD.CONVERSION'); OPEN
'&SAVEDLISTS&' TO SLFN ELSE CALL AVSS.ERROR(AV,'NOFILE','&SAVEDLISTS&');
OPEN 'UOM' TO UOMFN ELSE CALL AVSS.ERROR(AV,'NOFILE','UOM'); CALL
AVSS.PROCESS.START(AV,SHOW.TEXT,LABEL.TEXT); CALL
AVSS.PROCESS.DISPLAY(AV,SHOW.TEXT); ACCOUNT=AV<1,1,2>; CALL
AVSS.CONTROL.ACCOUNT.CHECK(AV,ACCOUNT,YN); IF YN='N' THEN; CALL
AVSS.PROCESS.DISPLAY(AV,~This process can only be run from the control account.');
GOTO 99; END; READ CARRIERITEM FROM MATCHUPCONFN,'CONWAY.CARRIER'
ELSE CARRIERITEM=''; READ CON.UOMITEM FROM
MATCHUPCONFN,'CONWAY.VOM' ELSE CON.UOMITEM=''; IF EMULATION#'WIV'
AND EMULATION#'WIT' AND EMULATION#'WWT' THEN; CALL
AVSS.PROCESS.DISPLAY(AV,'This process must be run a session using termite or viaduct.');
GOTO 99; END; * Prompt user to enter a drive and path of the shortage information; * on either
a floppy or hard disc.; CALL
AVSS.INPUT.CALL(AV,DOSLOC,'FREIGHT.BILL.UPLOAD-DOSLOC'," ); IF
DOSLOC='EXIT' THEN GOTO 99; MATCHSHUID=PORT:'+':EXEC.LEvEL;
ITEMCOUNT=0; * Must clear the file before beginning the process!!!; CALL
AVSS.EXECUTE(AV,'SELECT MATCHING.SHORTAGE.UPLOAD WITH ID =
"':MATCHSHUID:']"',", ",ITEMCOUNT); IF ITEMCOUNT THEN; CALL
AVSS.EXECUTE(AV,'CLEAR-FILE DATA MATCHING.SHORTAGE.UPLOAD,,'',,i,,,;
```

END; * Create the records for the Matching.shortage.upload tile. This; * is the temporary file to store the information that has been; * uploaded from disc.; ECHO ON; BEGIN CASE; CASE EMULATION='WIV'; TEXT=PCCMD:'KEY /P
<ALT+F>E<CTRL+BACK>':DOSLOC:'<CR><CTRL+BACK>MATCHING.SHORTAGE.UPLOAD<CR>':MATCHSHUID:'<CR>*<CR>M1440
P130<CR><CR><ALT>':ENDCMD; PRINT TEXT; EXECUTE "PCCTRL"; ECHO OFF; CASE EMULATION='WIT' OR EMULATION='WWT'; CALL PIX.CALL.DOS.PICK('MATCHING.SHORTAGE.UPLOAD ':MATCHSHUID:' FROM ':DOSLOC,'H',ERROR); IF ERROR=1 THEN; CALL AVSS.PROCESS.DISPLAY(AV,'The Termite transfer utility reported a fatal error and the transfer was not completed. Please consult your system administrator.'); GOTO 99; END; END CASE; * Begin the process of looping through temporary records to build; * the permanent MATCHING.SHORTAGE records.; CALL AVSS.PROCESS.DISPLAY(AV,VM:'Creating matching shortage records... ':VM:VM); CALL AVSS.EXECUTE(AV,'SELECT MATCHING.SHORTAGE.UPLOAD WITH ID = "':MATCHSHUID:'1"', "," ,"); LOOP; READNEXT MATCHSHUID ELSE MATCHSHUID=AM; UNTIL MATCHSHUID=AM DO; READ MATCHSHUITEM FROM MATCHSHUFN,MATCHSHUID ELSE MATCHSHUITEM=''; NUMLINES=DCOUNT(MATCHSHUITEM,AM); TOT=NUMLINES; ITEMCOUNT=0; FOR 1=1 TO NUMLINES; LINER=MATCHSHUITEM<I>; IF LINER#" THEN; PRO.NUMBER=TRIM(LINER[67,9]); EXCEPTION.TYPE=TRIM(LINER[129,4]); IF EXCEPTION.TYPE='INAC' THEN; LOCATE(PRO.NUMBER,PROLIsT,1;LOCVAL;'AR') ELSE; PROLIST=INSERT(PROLIST,1,LOCVAL;PRO.NUMBER); END; END ELSE; COMMODITY.LINE=TRIM(LINER[112,3]); CARRIER=TRIM(LINER[92,4]); UOM=TRIM(LINER[133,5]); * Assign the carrier code for the prefix of the; * Matching.shortage records;
LOCATE(CARRIER,CARRIERITEM,MATCHUPCON.CARRIER.CODE; LOCVAL ~ THEN; CARRIER.CODE=CARRIERITEM<MATCHUPCON.RECOV.CODE,LOCVAL>; END ELSE; ERROR.TEXT='The carrier code "':CARRIER:"' could not be converted, therefore Matching.shortage record could not be created - Process aborted!'; GOTO 99; END; * Read the Matching.shortage record Id from the Pro.number.line; * index file. This is to distinquish the different line number; * items. This must be done because the individual records with; * identical pro.numbers must be easily retrieved in the event; * that pro.number becomes inactive.; READV MATCHSHIDLIST FROM MATCHSHXPNFN,PRO.NUMBER,1 ELSE MATCHSHIDLIST="; COMMODITY.LINE.FOUND=0; IF MATCHSHIDLIST#" THEN;

NUMMATCHSHID=DCOUNT(MATCHSHIDLIST<1>,VM); FOR K=1 TO NUMMATCHSHID; MATCHSHID=MATCHSHIDLIST<1,K>; READ MATCHSHITEM FROM MATCHSHFN,MATCHSHID ELSE MATCHSHITEM="; LINENUM=MATCHSHITEM<MATCHSH.COMMODITY.LINE>; IF LINENUM=COMMODITY.LINE THEN; COMMODITY.LINE.FOUND=1; K=NUMMATCHBHID+5; END; NEXT K; END; IF COMMODITY.LINE.FOUND THEN; ORIGMATCHSHITEM=MATCHSHITEM; ACTION='E'; * Record is being updated and should have an Upload.date; * present. Thus, the Revision.date must be updated with the; * current date.; IF MATCHSHITEM<MATCHSH.UPLOAD.DATE>#" AND MATCHSHITEM<MATCHSH.UPLOAD.DATE>~DATE.STAMP THEN; LOCATE(DATE.STAMP,MATCHSHITEM,MATCHSH.REVISION.DATE;LOCVAL;'DR') ELSE;
MATCHSHITEM=INSERT(MATCHSHITEM,MATCHSH.REVISION.DATE,LOCVAL;DATE.STAMP); END; END; END ELSE; ORIGMATCHSHITEM="; MATCHSHITEM="; ACTION='A'; * Initiate Item and assign the record Id tor the; * Matching.shortage file.; NIID='*NI.':CARRIER.CODE; READVU NEXTITEM FROM MATCHSHDN,NIID,1 ELSE NEXTITEM=1;
CURRNEXTITEM=CARRIER.CODE:STR('0',IDLENGTH-LEN(NEXTITEM)):NEXTITEM; READ TEMPITEM FROM MATCHSHFN,CURRNEXTITEM THEN; TEMPITEM="; * AvCompiler; CALL AVSS.ERROR(AV,'FATAL','Matching Inventory number ':CURRNEXTITEM:' has already been issued !!!!'); RELEASE MATCHSHDN,NIID; GOTO 99; END; NEXTITEM=NEXTITEM+1; WRITEV NEXTITEM ON MATCHSHDN,NIID,1; MATCHSHID=CURRNEXTITEM; * Record is being created, therefore the upload.date musr be; * stamped since this is the original upload.;
MATCHSHITEM<MATCHSH.UPLOAD.DATE>=DATE.STAMP; END;
MATCHSHITEM<MATCHSH.SHIPPER.INFORMATION>=TRIM(LINERt1,30]);
MATCHsHITEM<MATCHsH.ORIGIN>=TRIM(LINER[31,3]);
MATCHSHITEM<MATCHSH.CONSIGNEE.INFORMATION>=TRIM(LINER[34,30]);
MATCHSHITEM<MATCHSH.DESTINATION>=TRIM(LINER[64,3]);
MATCHSHITEM<MATCHSH.PRO.NUMBER>=PRO.NUMBER;
MATCHSHITEM<MATCHSH.COMMODITY.LINE>=COMMODITY.LINE;
MATCHSHITEM<MATCHSH.SOMS.ID>=TRIM(LINER[76,16]);
MATCHSHITEM<MATCHSH.CARRIER>=CARRIER.CODE;
MATCHSHITEM<MATCHSH.COMMODITY.CODE>=TRIM(LINER[96,8]);

TEMPDATE=LINER [104,2] :'-':LINER [106,2]:'-':LINER[108,4];
MATCHSHITEM<MATCHSH.PRO.DATE> =ICONV(TEMPDATE,'D-');
NUMBER=TRIM(LINER[115,5]); GOSUB 100; * Convert number into "internal" format;
MATCHSHITEM<MATCHSH.QUANTITY> =NUMBER; NUMBER=TRIM(LINER
[120,9]); GOSUB 100; * Convert number into "internal" format;
MATCHSHITEM<MATCHSH.WEIGHT> =NUMBER;
MATCHSHITEM<MATCHSH.EXCEPTION.TYPE> =EXCEPTION.TYPE;
MATCHsHITEM<MATCHSH.DEsCRIPTION> =TRIM(LINER[138,67]); UOMID=UOM
[1,2]; READ UOM1TEM FROM UOMFN,UOMID THEN; UOMITEM=''; * Avgen compiler;
MATCHSHITEM<MATCHSH.UOM> =UOMID; END ELSE; IF NUM(UOM) THEN;
MATCHSHITEM<MATCHSH.UOM> ='BD'; END ELSE; TEMP.UOM=UOM[1,3]; IF
TEMP.UOM='HM/' THEN; SLASH.POS=INDEX(UOM,'/',1);
MATCHSHITEM<MATCHSH.UOM> =UOM[sLASH.POs+1,2]; END ELSE;
LOCATE(UOMID,CON.UOMITEM,MATCHUPCON.CARRIER.CODE;LOCVAL) THEN;
MATCHSHITEM<MATCHSH.UOM> =CON.UOMITEM<MATCHUPCON.RECOV.COD
E,LOCVAL>; END ELSE; MATCHSHITEM<MATCHSH.UOM> ='*BAD.DATA*'; END;
END; END; ND; PRONUM=MATCHSHITEM<MATCHSH.PRO.NUMBER>;
LOCATE(PRONUM,PRONUMLIST,1;LOCVAL;'AR') THEN; BADLIST<-1> =PRONUM;
END ELSE; PRONUMLIST=INSERT(PRONUMLIST,1,LOCVAL;PRONUM); END; CALL
AVSS.ITEM.UPDATE(AV,MATCHSHFN,MATCHSHDN,MATCHSHID,MATCHSHITEM,O
RIGMATCHESITEMACTION,'PROCESS','''); END; END; ITEMCOUNT=ITEMCOUNT+1;
IF REM(ITEMCOUNT,10)=0 THEN; PROGRESS.HEADING=SPACE(10):1TEMCOUNT:'
items completed, ':TOT-ITEMCOUNT:' items remaining':SPACE(5);
PROGRESS.HEADING<1,2> =SPACE(28):OCONV(ITEMCOUNT*100/TOT,'MR0'):'%';
CALL
AVSS.WINDOW.PROGRESS(AV,PROGRESS.WINID,PROGRESS.STATUS,PROGRESS.HE
ADING,ITEMCOUNT,TOT,'N','OFF'); END; NEXT I; CALL
AVSS.WINDOW.PROGRESS(AV,PROGRESS.WINID,'CLOSE',",",",'Y','ON'); CALL
AVSS.PROCESS.DISPLAY(AV,TOT:' records created and/or updated'); REPEAT; * Loop
through the PROLIST to close all of the inventory that requires; * an inactive status.; IF
PROLIST#" THEN; INACTIVE.RECORD.COUNT=0;
NUMPROLIST=DCOUNT(PROLIST<1>,VM); FOR 1=1 TO NUMPROLIST;
PRO.NUMBER=PROLIST<1,1>; READY MATCHLIST FROM
MATCHSHXPNFN,PRO.NUMBER,1 ELSE MATCHLIST=''; IF MATCHLIST#" THEN;

```
NUMMATCHLIST=DCOUNT(MATCHLIST<1>,VM); FOR J=1 TO NUMMATCHLIST;
MATCHSHID=MATCHLIST<1,J>; READ MATCHSHITEM FROM
MATCHSHFN,MATCHSHID ELSE MATCHSHITEM=";
ORIGMATCHSHITEM=MATCHSHITEM; ACTION='E';
MATCHSHITEM<MATCHSH.CLOSE.DATE>=DATE.STAMP; CALL
AVSS.ITEM.UPDATE(AV,MATCHSHFN,MATCHSHDN,MATCHSHID,MATCHSHITEM,;
ORIGMATCHSHITEM,ACTION,'PROCESS'," );
INACTIVE.RECORD.COUNT=INACTIVE.RECORD.COUNT+1; NEXT J; END; IF
INACTIVE.RECORD.COUNT=1 AND NOT(NOTE.PRINT) THEN; NOTE.PRINT=1; CALL
AVSS.PROCESS.DISPLAY(AV,VM:'Closing all inactive matching shortage records':VM);
END; NEXT I; IF INACTIVE.RECORD.COUNT>0 THEN CALL
AVSS.PROCESS.DISPLAY(AV,VM:'Closed ':INACTIVE.RECORD.COUNT:' inactive
matching short:; record(s)':VM); END; 99*; IF ERROR.TEXT#" THEN; CALL
AVSS.PROCESS.DISPLAY(AV,ERROR.TEXT); END; IF BADLIST#" THEN; WRITE
BADLIST ON SLFN,'UPLOAD.DUPELIST'; END; CALL
AVSS.PROCESS.END(AV,LABEL.TEXT); DATA AV; ENTER @AV.ENTER; STOP; 100*
Convert numbers to "internal" format; DECIMAL.COUNT=COUNT(NUMBER,'.'); IF
DECIMAL.COUNT=1 THEN; DECIMAL.POS=INDEX(NUMBER,';
NUMLEN=LEN(NUMBER); DECIMAL.LEN=NUMLEN-DECIMAL.POS;
CONV='MR':DECIMAL.LEN; NUMBER=ICONV(NUMBER,CONV); END; IF
DECIMAL.COUNT=0 THEN; NUMBER=ICONV(NUMBER,'MR0'); END; IF
DECIMAL.COUNT>1 THEN NUMBER='*BAD.DATA*'; RETURN; 1 );
AVPR.MATCHING.CONWAY.ORIGIN.UPLOAD; * Purpose :Upload matching origin
information from diskette for the carrier conway; * Status :5.80.A 07-11-97 16:21:55
1E4FOB70; * Notes :A) 07-11-97 FWN created; INPUT AV; INCLUDE AVIC AVIC.SETUP;
INCLUDE AV.EOUATE MATCHING.CARRIER.TERMINAL; INCLUDE AVMP
AVAL.EXEC.LEVEL; EMULATION=AV<1,1,3>; ENDCMD=CHAR(0);
LABEL.TEXT='Upload Carrier matching information from diskette'; NAME.POS=5;
PCCMD=CHAR(27):CHAR(8); PORT=AV<1,1,1>; PROGRESS.WINID='PROGRESS;
PROGRESS.STATUS='OPEN'; TOT=0; OPEN 'MATCHING.SHORTAGE.UPLOAD' TO
MSUFN ELSE CALL AVSS.ERROR(AV,'NOFILE,('MATCHING.SHORTAGE.UPLOAD);
OPEN 'DICT','MATCHING.CARRIER.TERMINAL' TO MCTDN ELSE CALL
AVSS.ERROR(AV,'NOFILE','D MATCHING.CARRIER.TERMINAL'); OPEN
'MATCHING.CARRIER.TERMINAL' TO MCTFN ELSE CALL
```

AVSS.ERROR(AV,'NOFILE','MATCHING.CARRIER.TERMINAL'); :ALL
AVSS.PROCESS.START(AV,SHOW.TEXT,LABEL.TEXT; .ALL
AVSS.PROCESS.DISPLAY(AV,SHOW.TEXT); IF EMULATION#'WIV' AND
EMULATION#'WIT' AND EMULATION#'WWT' THEN; CALL
AVSS.PROCESS.DISPLAY(AV,'This process must be run a session using termite or viaduct.');
GOTO 99; END; * Prompt user to enter a drive and path of the shortage information; On either a
floppy or hard disc.; CALL
AVSS.INPUT.CALL(AV,DOSLOC,'FREIGHT.BILL.UPLOAD-DOSLOC', "); IF
DOSLOC='EXIT' THEN GOTO 99; MATCHSHUID=PORT:'+':EXEC.LEVEL;
ITEMCOUNT=0; " Must clear the file before beginning the process!!!; CALL
AVSS.EXECUTE(AV,'SELECT MATCHING.SHORTAGE.UPLOAD WITH ID =
"':MATCHSHUID:'] "' , ", ", ITEMCOUNT); IF ITEMCOUNT THEN; CALL
AVSS.EXECUTE(AV,'CLEAR-FILE DATA MATCHING.SHORTAGE.UPLOAD','','','');
END; Create the records for the Matching.shortage.upload file. This; is the temporary file to
store the information that has been; uploaded from disc.; ECHO ON; BEGIN CASE; CASE
EMULATION='WIV'; TEXT=PCCMD:'KEY /P
<ALT+F>E<CTRL+BACK>':DOSLOC:'<CR><CTRL+BACK>MATCHING.SHORT
AGE.UPLOAD; <CR>':MATCHSHUID:'<CR>*<CR>M144O P;
O<CRxCR><ALT>':ENDCMD; PRINT TEXT; EXECUTE "PCCTRL"; ECHO OFF;
CASE EMULATION='WIT' OR EMULATION='WWT'; CALL
PIX.CALL.DOS.PICK('MATCHING.SHORTAGE.UPLOAD ':MATCHSHUID:' FROM
':DOSLOC,'H',ERROR); IF ERROR=1 THEN; CALL AVSS.PROCESS.DISPLAY(AV,'The
Termite transfer utility reported a fatal error and the transfer was not completed. Please; consult
your system administrator.'); GOTO 99; END; END CASE; Begin the process of Looping
through temp records to build the; permanent inventory shortage records.; CALL
AVSS.PROCESS.DISPLAY(AV,VM:'Creating matching origin records... ':VM:VM); CALL
AVSS.EXECUTE(AV,'SELECT MATCHING.SHORTAGE.UPLOAD WITH ID =
"':MATCHSHUID:'1"', " , ", "); LOOP; READNEXT MATCHSHUID ELSE
MATCHSHUID=AM; UNTIL MATCHSHUID=AM DO; READ MATCHSHUITEM FROM
MSUFN,MATCHSHUID ELSE MATCHSHUITEM=";
NUMLINES=DCOUNT(MATCHSHUITEM,AM); TOT=NUMLINES; ITEMCOUNT=0;
FOR 1=1 TO NUMLINES; LINER=MATCHSHUITEM<I>; IF LINER#" THEN;
MAX.LEN=LEN(LINER); MATCHCARTERMID=TRIM(LINERt1,NAME.POs-1]); READ
MATCHCARTERMITEM FROM MCTFN,MATCHCARTERMID THEN;

ORIGMATCHCARTERMITEM=MATCHCARTERMITEM; ACTION='E'; END ELSE; ORIGMATCHCARTERMITEM=''; MATCHCARTERMITEM=''; ACTION='A'; END; MATCHCARTERMITEM<MATCHCARTERM.NAME>=TRIM(LINER[NAME.POS,MAX. LEN-NAME.Pos+1]); ; CALL AVSS.ITEM.UPDATE(AV,MCTFN,MCTDN,MATCHCARTERMID,MATCHCARTERMITE M,ORIGMATCHCARTERMITEM,ACTION,'PROCESS', "); ITEMCOUNT=ITEMCOUNT+1; IF REM(ITEMCOUNT,100)=0 THEN; PROGRESS.HEADING=SPACE(10):1TEMCOUNT:' items completed, ':TOT-ITEMCOUNT:' items remaining':SPACE(5); PROGREsS.HEADING<1,2>=sPACE(28):0CONV(ITEMCOUNT*100/TOT,'MRO'):'%'; CALL AVSS.WINDOW.PROGRESS(AV,PROGRESS.WINID,PROGRESS.STATUS,PROGRESS.HE ADING,ITEMCOUNT,TOT,'N','OFF'); END; END; NEXT I; REPEAT; CALL AVSS.WINDOW.PROGRESS(AV,PROGRESS.WINID,'CLOSE','','','',' T' ,'UN' 1; CALL AVSS.PROCESS.DISPLAY(AV,TOT:' items completed'); 99*; CALL AVSS.PROCESS.END(AV,LAGEL.TEXT); DATA AV; INTER @AV.ENTER; STOP; SUBROUTINE AVSU.L.CARRMAT(AV); * Purpose : took subroutine for the carrier matching file; * Status :5.80.A 12-04-95 16:17:19 EECD388; * Notes :A) 12-04-95 FH3 upgrade to 5.80; * Argument :1) AV [P] Avexxis system info; NCLUDE AVIC AVIC.AVUP; NCLUDE AV.EQUATE CARRIER.MATCHING; Get the carrier file names, open files; CARRIER.FILENAME='OSD.':ITEM<CARRMAT.CARRIER>; CARRIER.FINDFILENAME=CARRIER.FILENAME:'.FIND'; OPEN CARRIER.FILENAME TO XXFN ELSE CALL AVSS.ERROR(AV,'NOFILE',CARRIER.FILENAME); OPEN CARRIER.FINDFILENAME TO XXFINDFN ELSE CALL AVSS.ERROR(AV,'NOFILE',CARRIER.FINDFILENAME); OPEN 'DICT',CARRIER.FINDFILENAME TO XXFINDDN ELSE CALL AVSS.ERROR(AV,'NOFILE',CARRIER.FINDFILENAME); * Build the find string from the current item; FINDSTRING=ITEM<CARRMAT.FREIGHT.BILL.NUMBER>:' ':ITEM<CARRMAT.DESCRIPTION>:' ':ITEM<CARRMAT.MANUFACTURER>:' ':ITEM<CARRMAT.PA; RT.NUMBER>:' ':ITEM<CARRMAT.SERIAL.NUMBER>:' ':ITEM<CARRMAT.SHIPPER>:' ':ITEM<CARRMAT.CONSIGNEE>; FINDSTRING=TRIM(FINDSTRING); CONVERT VM TO ' ' IN FINDSTRING; IF FINDSTRING='' THEN; DISPLAY.MESSAGE='At least one field must be filled in before searching.'; CALL AVSS.DISPLAY.MESSAGE(AV,DISPLAY.MESSAGE,REPAINT.LINES);

CALL AVUP.REPAINT(AV); RETURN; END; w CalL tind, get matcn;
FINDATTLIST=1:5VM:2:5VM:3:5VM:4:5VM:5:5VM:6:5VM:7; DEPTH=3;
REPAINT='N'; LOOK.CHOICE= "; CALL
AVSS.FIND(AV,FINDSTRING,XXFINDFN,XXFINDDN,XXFN,FINDATTLIST,DEPTH,LO
OK.CHOICE, ",'AVUP',REPAINT,'ON'); IF REPAINT#'N' THEN
REPAINT.LINES=INSERT(REPAINT.LINES,-1;REPAINT); IF LOOK.CHOICE#'EXIT'
THEN; ITEM<CARRMAT.MATCH>=LOOK.CHOICE; REPAINT.LINES=3:VM:24;
CALL AVUP.REPAINT(AV); END; 99*; RETURN; AVSU.S.CARRMAT; SUBROUTINE
AVSU.S.CARRMAT(AV); * Purpose :Start subroutine for the carrier matching file; * Status
:5.80.A 12-04-95 16:20:27 1CEB8A28; * Notes :A) 12-04-95 FH3 upgrade to 5.80; * Argument
:1) AV [P] Avexxis system info; INCLUDE AVIC AVIC.AVUP; INCLUDE AV.EQUATE
CARRIER.MATCHING; INCLUDE AV.EQUATE INVENTORY; IF MODE='NEW' THEN;
READ INVENITEM FROM IDREFFN,ID ELSE INVENITEM=";
ITEM<CARRMAT.CARRIER>=INVENITEM<INVEN.CARRIER>;
ITEM<CARRMAT.FREIGHT.BILL.NUMBER>=INVENITEM<INVEN.PRO.NUMBER>;
ITEM<CARRMAT.DESCRIPTION>=INVENITEM<INVEN.PRICING.DESCRIPTION>;
ITEM<CARRMAT.MANUFACTURER>=INVENITEM<INVEN.MANUFACTURER>;
END; 99*; RETURN; AVSU.Z.INVEN.PGU.UPDATE; SUBROUTINE
AVsU.Z.INVEN.PGU.UPDATE(AV); *Purpose : Inventory - update PGU info; *Status :5.80.G
04-13-98 06:40:23 7C4AEB7F; *Notes :A) 12-04-95 FH3 upgrade to 5~80>>:B) 10-30-96
BAR add Rsearch to skip list>>:C) 01-02-98 NDL Update *Matching.,inventorypgu and qty
changes>>:D) 01-04-98 RAR close files opened>>:E) 02-26-98 NDL add code for pgu
*datrash log>>:F) 04-01-98 NDL put mu Lt.carr and datrash into prod>>:G) 04-13-98 NDL
add account to datrash; *Argument :1) AV lP] Avexxis system info; INCLUDE AVIC
AVIC.AVUP; INCLUDE AV.EQUATE MATCHING. INVENTORY; [NCLUDE
AV.EQUATE PORTABLE.GROUP; INCLUDE AV.EOUATE UNLOAD; OPEN
'MATCHING.INVENTORY' TO MATCHINVENFN ELSE CALL
AVSS.ERROR(AV,'NOFILE,>MATCHING.INVENTORY); OPEN
'MATCHING.INVENTORY.INDEX.INVENTORY' TO MATCHINVENXINVENFN ELSE
CALL AVSS.ERROR(AV,'NOFILE','MATCHING.INVENTORY.INDEX.INVENTORY;
SKIPLIST='MARKING':AM:'STOREPR':AM:'RSEARCH'; CALL
AVSS.FILE.NAME(AV,'PORTABLE.GROUP',PORTABLE.GROUP.ACCOUNT,");
NUMLINES=DCOUNT(ORIGITEM<UNLOAD.QTY.ON.HAND>,VM); CHGLIST=";
ITEMCHECK="; FOR 1=1 TO NUMLINES;

```
SOURCEITEM=ORIGITEM<UNLOAD.PORTABLE.GROUP,I>;
LOCATE(SOURCEITEM,SKIPLIST;LOCATT) THEN SOURCEITEM="; IF
SOURCEITEM#" THEN;
LOCATE(SOURCEITEM,ITEM,UNLOAD.PORTABLE.GROUP;TEMPVAL) THEN;
ITEMCHECK<1,TEMPVAL>='X';
DIFF=ITEM<UNLOAD.QTY.ON.HAND,TEMPVAL>-ORIGITEM<UNLOAD.QTY.ON.H
AND,I>; IF DIFF#0 THEN; CHGLIST<1,-1>=SOURCEITEM; CHGLIST<2,-1>=DIFF;
END; END ELSE; CHGLIST<1,-1>=SOURCEITEM;
CHGLIST<2,-1>=ORIGITEM<UNLOAD.QTY.ON.HAND,1>*(-1); END; END; NEXT I;
NUMLINES=DCOUNT(ITEM<UNLOAD.PORTABLE. GRUUR>, VM'; FOR 1=1 TO
NUMLINES; IF ITEMCHECK<1,1>#'X' THEN; IF
ITEM<UNLOAD.PORTABLE.GROUP,I>#" THEN;
CHGLIST<1,-1>=ITEM<UNLOAD.PORTABLE.GROUP,I>;
CHGLIST<2,-1>=ITEM<UNLOAD.QTY.ON.HAND,I>; END; END; NEXT 1; IF
CHGLIST#" THEN; NUMLINES=DCOUNT(CHGLI ST<1 >,VM); FOR 1=1 TO
NUMLINES; PGUID=CHGLIST<1,1>; LOCATE(PGUID,SKIPLIST;LOCATT) ELSE;
LOOP; LOCKFLAG=' ON'; CALL
AVSS.LOCK(AV,AVLOCKFN,PORTABLE.GROUP.ACCOUNT,'PORTABLE.GROUP',PGUI
D,LOCKFLAG," ); WHILE LOCKFLAG='LOCKED' DO; DISPLAY.MESSAGE=,portable
Group ':PGUID:' is locked; processing on this Inventory item cannot be completed until this item
is locked.'; CALL AVSS.DISPLAY.MESSAGE(AV,DISPLAY.MESSAGE,REPAINT.LINES);
REPEAT; * code to log if pguid = datrash; IF PGUID=''DATRASH'' THEN;
QUANTITY=CHGLIST<2,1>; CALL
AVSU.Z.ALL.PGU.DATRASH.LOG(AV,ID,QUANTITY," ); END; READU PGUITEM
FROM REFFN(UNLOAD.PORTABLE.GROUP),PGUID THEN;
NUMBER.ITEMS.ON.PGU=DCOUNT(PGUITEM<PGU.INVENTORY>,VM); IF
NUMBER.ITEMS.ON.PGU+0=0 AND PGUITEM<PGU.CARRIER>=" AND
PGUITEM<PGU.MULTIPLE.CARRIERS>='N' THEN; CARRIER.ID=ID[1,2];
PGUITEM<PGU.CARRIER>=CARRIER.ID; END;
LOCATE(ID,PGUITEM,PGU.INVENTORY;TEMPVAL) THEN;
PGUITEM<PGU.QUANTITY,T,EMPVAL>=PGUITEM<PGU.QUANTITY,TEMPVAL>
+CHGLIST<2,1>; IF PGUITEM<PGU.QUANTITY,TEMPVAL>=0 THEN;
PGUITEM=DELETE(PGUITEM,PGU.QUANTITY,TEMPVAL);
PGUITEM=DELETE(PGUITEM,PGU.INVENTORY,TEMPVAL); END; END ELSE;
```

PGUITEM<PGU.INVENTORY,-1>=ID;
PGUITEM<PGU.QUANTITY,-1>=CHGLIST<2,1>; END; CALL
AVSU.Z.ALL.PGU.CLEAR.CARRIER(AV,PGUITEM); WRITE PGUITEM ON
REFFN(UNLOAD.PORTABLE.GROUP),PGUID; END ELSE; RELEASE
REFFN(UNLOAD.PORTABLE.GROUP),PGUID; END; END; Update the matching. inventory
PGU and qty; READV MATCHINVENID FROM MATCHINVENXINVENFN,ID,1 ELSE
MATCHINVENID=''; READ MATCHINVENITEM FROM
MATCHINVENFN,MATCHINVENID THEN;
LOCATE(CHGLIST<1,1>,MATCHINVENITEM,MATCHINVEN.PORTABLE.GROUP;FIN
DVAL) THEN;
MATCHINVENITEM<MATCHINVEN.PORTABLE.GROUP.QOH,FINDVAL>=MATCHI
NVENITEM<MATCHINVEN.PORTABLE.GROUP.QOH,FINDVAL>+CHGLIST<2,1>;
IF MATCHINVENITEM<MATCHINVEN.PORTABLE.GROUP.QOH,FINDVAL>=0
THEN;
MATCHINVENITEM=DELETE(MATCHINVENITEM,MATCHINVEN.PORTABLE.GROUP
.QOH,FINDVAL);
MATCHINVENITEM=DELETE(MATCHINVENITEM,MATCHINVEN.PORTABLE.GROUP
,FINDVAL);
MATCHINVENITEM=DELETE(MATCHINVENITEM,MATCHINVEN.PORTABLE.GROUP
.QTY.COMMITTED,FINDVAL); END; END ELSE;
MATCHINVENITEM<MATCHINVEN.PORTABLE.GROUP,-1>=CHGLIST<1,1>;
MATCHINVENITEM<MATCHINVEN.PORTABLE.GROUP.QOH,-1>=CHGLIST<2,1>;
END; WRITE MATCHINVENITEM ON MATCHINVENFN,MATCHINVENID; END; NEXT
I; END; 99*; CLOSE MATCHINVENFN; CLOSE MATCHINVENXINVENFN;
NUMLINES=DCOUNT(CHGLIST<1>,VM); FOR 1=1 TO NUMLINES;
PGUID=CHGLIST<1,1>; CALL
AVSS.LOCK(AV,AVLOCKFN,PORTABLE.GROUP.ACCOUNT,'PORTABLE.GROUP',PGUI
D,OFF,''); NEXT I; RETURN; END; AVSU,P.MATCHSESA.1; SUBROUTINE
AVsU.P.MATCHSESA.1(AV); * Purpose :Postsub routine for the search.matching.standalone
file; * Status :5.80.B 07-01-97 14:26:15 D1817A5; * Notes :A) 06-26-97 FWN create>>:B)
07-01-97 FWN move to live; * Argument :1) AV [P] Avexxis system info; INCLUDE AVIC
AVIC.AVUP; INCLUDE AV.EQUATE MATCHING.SEARCH.STANDALONE; IF
NOT(TEXTCHG) THEN RETURN; BEGIN CASE; CASE
CURRATT=MATCHSESA.QUANTITY.FROM; IF

ITEM<MATCHSESA.QUANTITY.TO>=" THEN;
ITEM<MATCHSESA.QUANTITY.TO>=ITEM<MATCHSESA.QUANTITY.FROM>;
END; CASE CURRATT=MATCHSESA.PRO.NUMBER.DATE.FROM; IF
ITEM<MATCHSESA.PRO.NUMBER.DATE.TO>=" THEN;
ITEM<MATCHSESA.PRO.NUMBER.DATE.TO>=ITEM<MATCHSESA.PRO.NUMBER.
DATE.FROM>; END ,; CASE CURRATT=MATCHSESA.WEIGHT.FROM; IF
ITEM<MATCHSESA.WEIGHT.TO>=" THEN;
ITEM<MATCHSESA.WEIGHT.TO>=ITEM<MATCHSESA.WEIGHT.FROM>; END;
CASE CURRATT=MATCHSESA.SHORTAGE.RECORD.CREATE.FROM; IF
ITEM<MATCHSESA.SHORTAGE.RECORD.CREATE.TO>= " THEN;
ITEM<MATCHSESA.SHORTAGE.RECORD.CREATE.TO>=ITEM<MATCHSESA.SHOR
TAGE.RECORD.CREATE.; FROM>; END; END CASE; 99*; RETURN;
AVSU.S.MATCHINVEN; SUBROUTINE AVSU.S.MATCHINVEN(AV) ·; * Purpose :Start for
Matching.Inventory; * Status :6.00.N 06-24-98 12:45:20 84609DCF; * Notes :A) 01-11-97 RAR
initial version>>:B) 07-01-97 FWN move to live>>:C) 07-03-97 RAR add imageid>>:D)
07-15-97 FWN added ; Temp.image to comman variable list>>:E) 07-23-97 FWN added code
to change the Inventory field to either "Blocked" or 'lBlocked/Rec; uired">>:F) 07-28-97 FWN
added code to lock the inventory record in the "Open/View" mode>>:G) 07-30-97 FWN
changed the carrier ; field to 'BR' when mode is not new>>:H) 08-01-97 FWN added code that
rebuilds portable.group section>>:!) 08-02-97 FWN added section; to build
ORIGPGULIST>>:J) 08-02-97 FWN fixed qty that is obtained from the inventory file>>:K)
08-07-97 RAR Rearrange include; file>>:L) 08-17-97 RAR use C.INVENITEM>>:M)
08-18-97 BAR add input.by fill,,:N) 06-24-98 RAR add blinking product image; * Argument :1)
AV [P] Avexxis system info -; INCLUDE AVSU.RECOV AVSU.N.MATCHINVEN;
INCLUDE AV.EQUATE INVENTORY; INCLUDE AV.EQUATE MATCHING.
INVENTORY; INCLUDE AV.EQUATE AV.USER; INCLUDE AVMP AVAL.EXEC.LEVEL;
C.IMAGEID='‘; :.ORIGPGULIST="-C.ORIGPGULIST='‘; C.RENAME.IMAGE=0;
C.TEMP.IMAGE="; DRIGPGULIST="; WSID=AV<1,1,1>:'+':EXEC.LEVEL;
WSITEM="; CALL AVSS.WORKSPACE(AV,WSID,WSITEM,'OFF'); BEGIN CASE; CASE
WSITEM='LOOKUP'; C.MATCHING.EDIT=0; CASE WSITEM=";
C.MATCHING.EDIT=1; CASE WSITEM#"; C.MATCHING.EDIT=1;
C.COMMODITY.CODE=WSITEM<1>; C.MATCHING.DICTIONARY=WSITEM<2>;
ITEM<MATCHINVEN.COMMODITY.CODE>=C.COMMODITY.CODE; END CASE; F
MODE#'NEW' THEN; * Check for image and display on screen; CODE=0;

IMAGEID=ID[3,987654321]; SUBDIRECTORY=ITEM<MATCHINVEN.CARRIER>:'1';
CALL PIX.EXISTS(C.IMAGE.PATH:SUBDIRECTORY:IMAGEID:'.BMP',CODE); IF CODE
THEN; ITEM<MATCHINVEN.PRODUCT.IMAGE>='IMAGE'; END ELSE;
ITEM<MATCHINVEN.PRODUCT.IMAGE>=''; END;
REQBLocK<1,MATCHINVEN.INVENTORR>=,BR,;
REQBLOCK<1,MATCHINVEN.CARRIER>='BR';
INVENID=ITEM<MATCHINVEN.INVENTORY>; IF MODE='OPEN' THEN;
LOCKFLAG='ON'; PORT=AV<1,1,1>; PASS.PORTEXEC=PORT:'+':EXEC.LEVEL:
CALL AvSS.FILE.NAME(AV,'INVENTORY',INVENTORY.ACCOUNT,''); CALL
AVSS.LOCK(AV,AVLOCKFN,INVENTORY.ACCOUNT,'INVENTORr',INVENID,;
LOCKFLAG,PASS.PORTEXEC); IF LOCKFLAG='LOCKED' THEN;
DISPLAY.MESSAGE='The inventory record ':INVENID:' was locked, therefore the
Matching.inventory record could not be opened'; PROBLEM.FOUND=1; GOTO 99; END;
END; Clear the Portable group section!!!;
NUMLINES=DCOUNT(ITEM<MATCHINVEN.PORTABLE.GROUP>,VM); CALL
AVUP.COLUMN.VALUE.ADJUsT(AV,3,'D',1,NUMLINEs); IF INVENID#'' THEN; READ
C.INVENITEM FROM C.INVENFN,INVENID ELSE C.INVENITEM='';
ITEM<MATCHINVEN.PORTABLE.GROUP>=C.INVENITEM<INVEN.PORTABLE.GRO
UP>; CALL AVUP.FILL(AV,MATCHINVEN.PORTABLE.GROUP,",",");
ITEM<MATCHINVEN.PORTABLE.GROUP.QOH>=C.INVENITEM<INVEN.QTY.ON.H
AND>; CALL AVUP.FILL(AV,MATCHINVEN.PORTABLE.GROUP.QOH,",","");
ITEM<MATCHINVEN.PORTABLE.GROUP.QTY.COMMITTED>=C.INVENITEM<INV
EN.QTY.COMMITTED>; CALL
AVUP.FILL(AV,MATCHINVEN.PORTABLE.GROUP.QTY.COMMITTED,",", "); END; *
Build ORIGPGULIST with the original inventory amounts then pass; * variable in common to be
used in the wrapup to verify that the; * total PGU quantity has not been changed.;
ORIGPGULIST<1>=1TEM<MATCHINVEN.PORTABLE.GROUP>;
ORIGPGULIST<2>=ITEM<MATCHINVEN.PORTABLE.GROUP.QOH>;
ORIGPGULIST<3>=1TEM<MATCHINVEN.PORTABLE.GROUP.QTY.COMMITTED>;
C.ORIGPGULIST=ORIGPGULIST; END ELSE;
REQBLOCK<1,MATCHINVEN.INVENTORY>='R';
REQBLOCK<1,MATCHINVEN.CARRIER>='R'; END; * Fill Input.by attribute with
employee number from av.user; IF ITEM<MATCHINVEN.INPUT.BY>='' THEN; OPEN
'AV.USER' TO AVUSRFN ELSE CALL AVSS.ERROR(AV,'NOFILE','AV.USER');

USER.ACCOUNT=AV<1,1,2>; READY PERSONNEL.NUMBER FROM AVUSRFN,USER.ACCOUNT,AVUSR.PERSONNEL.REFERENCE ELSE PERSONNEL.NUMBER=";
ITEM<MATCHINVEN.INPUT.BY>=PERSONNEL.NUMBER; END; 99*; RETURN; AVSU.E. MATCHINVEN; SUBROUTINE AVSU.E.MATCHINVEN(AV); Purpose :Start for Matching.Inventory; Status :6.00.G 08-07-97 15:02:24 36F4F7CB; Notes :A) 01-11-97 RAR initial version>>:B) 03-10-97 RAR initial version>>:C) 07-01-97 FWN move to ; live>>:D)07-03-97 RAR USE; c.imageid>>:E) 07-15-97 FWN included new common variable C.TEMP.IMAGE>>~:F) 07-28-97 FWN add code to unlock inventory record>>:G); 08-07-97 RAR Rearrange include file; * Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV AVSU.N.MATCHINVEN; INCLUDE AV.EQUATE MATCHING. INVENTORY; INCLUDE AVMP AVAL.EXEC.LEVEL; IF MODE='NEW' THEN; * Check for Termite for Windows; * PIX.TERMITE.INFO returns information about the current version of; * TERMITE this routine is run from.; * STAT<1> = 1 if TERMITE for Windows running or 0 if DOS version; * STAT<2> = 1 if current PC is color, 0 if mono; * STAT<3> = 1 if blinking is enabled on PC, 0 if not; * STAT<4> = 1 if this PC has a mouse that TERMITE can use, 0 if not; STAT="; TERMITE.OK=0; IALL PIX.TERMITE.INFO(STAT); IF STAT<1>='1' THEN; TERMITE.OK=1; END; IF TERMITE.OK THEN; * Get image filename/number; IMAGE.FILE.NAME=C.IMAGEID:'.BMP'; IF IMAGE.FILE.NAME#" THEN NULL: * For Avgen compiler only!!!; * Check to see if image exists; NO.IMAGE=1; CODE=0; CALL PIX.EXISTS(C.IMAGE.PATH:IMAGE.FILE.NAME,CODE); IF CODE=1 THEN NO.IMAGE=0; IF NOT(NO.IMAGE) THEN; CALL PIX.ERASE.DOS.FILE(C.IMAGE.PATH:C.TEMP.IMAGE); PRINT NOTEPRINT:'Image Deleted !'; SLEEP 1; NOTE="; PRINT NOTEPRINT:NOTE; END; END; END; INVENID=ITEM<MATCHINVEN.INVENTORY>; IF INVENID#" THEN; PORT=AV<1,1,1>; PASS.PORTEXEC=PORT:'+':EXEC.LEVEL; CALL AVSS.FILE.NAME(AV,'INVENTORT',I NVENTORT.ACCOUNT, ' '); CALL AVSS.LOCK(AV,AVLOCKFN, INVENTORY.ACCOUNT, ' INVENTORY' , INVENID, 'OFF' ,PASS.PORTEXEC); END; 99*; RETURN; AVSU.W.MATCHSE; SUBROUTINE AVSU.W.MATCHSE(AV); Purpose :; Status :5.86.A 07-01-97 14:14:01 19559679; Notes :A) 07-01-97 FWN move to live; Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV AVSU.N.MATCHSE; INCLUDE AV. EQUATE MATCHING . SEARCH; INCLUDE AVMP AVAL.EXEC.LEVEL; r ***** * * **** * * ** *** * *** * * * * * * ****** * ** ** ** ****** **** ********* *** ***; r** Any error traping needs to be

performed before this point ***; r * * ** * * * ***** * * * *** * * ** * * * * * * * * **** * * * *** * * * **** * * ** * * ***** *** ** * ***; * Stamp carrier commodity code; IF MODE='NEW' THEN;
ITEM<MATCHSE.CARRIER.COMMODITY>=ITEM<MATCHSE.CARRIER>:'+':C.COMMODITY.CODE; END; Restamp commodity code from origitem with carrier code from item.; This code is used when a record is edited from the Matching.search; dictionary.; IF ORIGITEM<MATCHSE.CARRIER>#ITEM<MATCHSE.CARRIER> AND ORIGITEM<MATCHSE.CARRIER; >#''THEN;
ITEM<MATCHSE.CARRIER.COMMODITY>=ITEM<MATCHSE.CARRIER>:'+':FIELD(ITEM<MATCHSE.CARRIER.COMMODITY>,'+',2); END; * Assign nextitem counter manually - To support multiple dicts; CALL AVSS.NEXTITEM(AV,C.MATCHSEDN,'',ID; CURRNEXTITEM=ID; 99*; RETURN; AVSU,L.MATCHINVEN.2; SUBROUTINE AVSU.L.MATCHINVEN.2(AV); *Purpose; *Status :6.00.B OB-07-97 15:UZ:ZS SZ/UD~BE; *Notes:A) 07-01-97 FWN move to live>>:B) 08-07-97 RAR Rearrange include file; * Argument :1) AV [P] Avexxis system info; INCLUDE AVSU.RECOV AVSU.N.MATCHINVEN; INCLUDE AV.EQUATE MATCHING. INVENTORY; INVENID=ITEM<MATCHINVEN.UNMATCHED.SEARCH,CURRVAL>;
EXECTEXT='AV UP ':C.MATCHING.DICTIONARY:':MATCHING.SEARCH VIEW ':INVENID; CALL AVSS.EXECUTE(AV,EXECTEXT,'N',","); 99*; RETURN;
ACTION='VIEW.IMAGE; PROMPT=' View image';
LOCATE(ACTION,ACTION.LIST.2,1;LOCVAL;"AL") ELSE;
PROMPT.LIST.2=INSERT(PROMPT.LIST.2,1,LOCVAL;PROMPT);
ACTION.LIST.2=INSERT(ACTION.LIST.2,1,LOCVAL;ACTION); END; IF C.MATCHING.EDIT THEN; * Re-capture image and replace existing image;
ACTION='OVERWRITE.IMAGE'; PROMPT=' Overwrite Image';
LOCATE(ACTION,ACTION.LIST.2,1;LOCVAL;"AL") ELSE;
PROMPT.LIST.2=INSERT(PROMPT.LIST.2,1,LOCVAL;PROMPT);
ACTION.LIST.2=INSERT(ACTION.LIST.2,1,LOCVAL;ACTION); END; Delete existing image; ACTION='DELETE.IMAGE'; PROMPT=' Delete image';
LOCATE(ACTION,ACTION.LIST.2,1;LOCVAL;"AL") ELSE;
PROMPT.LIST.2=INSERT(PROMPT.LIST.2,1,LOCVAL;PROMPT);
ACTION.LIST.2=INSERT(ACTION.LIST.2,1,LOCVAL;ACTION); END; END; END; IF PROMPT.LIST.2=" THEN GOTO 99;
AVMENUID.2='AVUP.CHOOSE.':PORT:'.':EXEC.LEVEL;

BASEAVMENUID.2='AWP.CHOOSE.TEMPLATE.PORT'; HEAD.TEXT='Select option':VM; CALL AVSS.MENU.FILL(AV,BASEAVMENUID.2,AVMENUID.2,HEAD.TEXT,PROMPT.LIST.2, ACTION.LIST.2,HELP.LIST.2); LOOP; REPAINT='N'; CURSOR.STATUS='ON'; CALL AVSS.MENU(AV,'1',BASEAVMENUID.2,CHOICER,REPAINT,CURSOR.STATUS,SCREENPAINT); UNTIL CHOICER='EXIT' DO; BEGIN CASE; CASE CHOICER='CAPTURE.NEW.IMAGE'; PIX.WIN.RUN runs a Windows program from TERMITE; and uses the value of STATE to setup how it's run.; STATE = 1 - Activates and displays the Window; STATE = 2 - Activates and minimizes the Window; STATE = 3 - Activates and maximizes the Window; STATE = 4 - Displays the window minimized but; does not change active window; NOTE='Capturing image ...'; PRINT NOTEPRINT:NOTE; PROGID='C:\SNAPPY\SNAPPY.EXE'; PROGID=PROGID; STATE='3'; STATE=STATE; SUCCESSFUL=''; CALL PIX.WIN.RUN(PROGID,STATE,SUCCESSFUL); IF NOT(SUCCESSFUL) THEN; DISPLAY.MESSAGE='TERMITE could not successfully run the image capturing software. Make sure It is not already; running on the desktop and then consult your system administrator.'; PROBLEM.FOUND=1; GOTO 99; END; APP.NAME='Snappy Video Snapshot'; KEYS='SP'; GOSUB 400; *Sleep while image is painted on screen; SLEEP 25; * Send carriage return to remove full screen snapshot; APP.NAME='Snappy Snapshot'; KEYS='CR'; GOSUB 400; * Sleep while PC receives carriage returns; SLEEP 5; * Send keys to select save; NOTE='Saving image ....'; PRINT NOTEPRINT:NOTE; APP.NAME='Snappy - Image1'; KEYS="'S"'; GOSUB 400; Assign new temporary ID; IF MODE='NEW' THEN; CURRTIME=TIME(); IF CURRTIME>999 THEN; CURRTIME=CURRTIME[LEN(CURRTIME)-3,4]; END; C.IMAGEID='T':AV<1,1,1>'R°/3':CURRTIME 'R%.4'; C.TEMP.IMAGE=C.IMAGEID:'.BMP'; END; Send name of file; SLEEP 2; APP.NAME='Save Snappy Image'; KEYS="":C.IMAGE.PATH: SUBDI RECTORT :C.IMAGEID:',BMP''CR'; GOSUB 400; * Close Snappy; SLEEP 4; NOTE='Closing Snappy'; PRINT NOTEPRINT:NOTE; APP.NAME='Snappy - ':C.IMAGEID; KEYS='A4'; GOSUB 400; APP.NAME='Snappy - ':C.IMAGEID:'.BMP'; KEYS='A4'; GOSUB 400; Check to see if Image is really on Disk before; flipping rename flag; IF MODE='NEW' THEN; CODE=0; CTR=0; LOOP; CTR=CTR+1; UNTIL CTR>10 OR CODE=1 DO; CALL PIX.EXISTS(C.IMAGE.PATH:SUBDIRECTORY:C.IMAGEID:'.BMP',CODE); SLEEP 1; REPEAT; IF CODE=1 THEN; C.RENAME.IMAGE=1; END; END; RUNTIME=0; TIMEOUT=35; LOOP; SLEEP 1; RUNNING=''; CALL

PIX.WIN.RUNNING(APP.NAME,RUNNING); RUNTIME=RUNTIME+1; UNTIL NOT(RUNNING) OR RUNTIME>TIMEOUT DO REPEAT; NOTE="; PRINT NOTEPRINT:NOTE; GOTO 99; CASE CHOICER='DELETE.IMAGE'; DELETE.COMMAND="; CALL AVSS.WINDOW.EDIT(AV,'AVUP.PROMPT','Enter DELETE to remove the image for this record ? ','Delete WARNING !',' ,DELETE.COMMAND,'6','N','ON'); IF OCONV(DELETE.COMMAND,'MCU')='DELETE' THEN; GOSUB 500; CALL PIX.ERASE.DOS.FILE(C.IMAGE.PATH:SUBDIRECTORY:C.IMAGEID:'.BMP'); PRINT NOTEPRINT:'lmage Deleted !'; SLEEP 1; NOTE=" -; PRINT NOTEPRINT:NOTE; IF MODE='NEW' THEN C.IMAGEID="; END; GOTO 99; CASE CHOICER='VIEW.IMAGE'; GOSUB 300; * View Snappy image; GOTO 99; END CASE; REPEAT; END; END CASE; REPEAT; END; 99*; RETURN; 100* Verify that application is being run from Termite; Check for Termite for Windows; * PIX.TERMITE.INFO returns information about the current version of; * TERMITE this routine is run from.; STAT<1> = 1 if TERMITE for Windows running or O if DOS version; STAT<2> = 1 if current PC is color, 0 if mono; STAT<3> = 1 if blinking is enabled on PC, 0 if not; STAT<4> = 1 if this PC has a mouse that TERMITE can use, 0 if not; STAT=''; TERMITE.OK=0; CALL PIX.TERMITE.INFO(STAT); IF STAT<1>='1' THEN; TERMITE.OK=1; END; RETURN; 200* ChecK tor existing image; NO.IMAGE=1; CODE=0; GOSUB 500; CALL PIX.EXISTS(C.IMAGE.PATH:SUBDIRECTORY:C.IMAGEID:'.BMP',CODE); IF CODE=1 THEN NO.IMAGE=0; ACT ION . LlsT .2="; PROMPT . L l ST . 2= "; HELP.LIST.2="; RETURN; 3OO* View a Snappy image; * Define a secondary window using Termite AIF sequence; GOSUB 500; PRINT ESC:' 29;1;8;22;8;32;9;34;;;;1;1wMatchimageWindow;':ESC:'\'; Display image in secondary window using termite AIF sequence; IF MODE='NEW' THEN; PRINT ESC:' 32;1;1;8;32;2wMatchimage;file=':C.IMAGE.PATH:C.TEMP.IMAGE:ESC:'\'; END ELSE; PRINT ESC:' 32;1;1;8;32;2wMatchimage;file=':C.IMAGE.PATH:SUBDIRECTORY:C.IMAGEID:'.BMP':ESC:'\'; END; Display Avgen window behind the bitmap image to ser up a promp˜; for an escape sequence.; REPAI NT='N'; CURSOR.STATUS='ON'; CALL AVSS.WINDOW.TEXT(AV,'MATCHING.IMAGE',",",REPAINT,CURSOR.STATUS); IF REPAINT#'N' THEN REPAINT.LINES<-1>=REPAINT; * After the user presses ESC to exit the Avgen window (Which he can't; * see) destroy the secondary window using the Termite AIF sequence; PRINT ESC:' 29;2wMatchimageWindow':Esc:\'; PRINT ESC:'

37;1;1wMatchimage':ESC:'\'; PRINT ESC:' 10;2wMatchimage':ESC:'\'; PRINT ESC:' 28;3w':ESC:'\'; RETURN; 400* Send keys; PRINT CHAR(27):' k':APP.NAME:'%':KEYS:CHAR(27):'\'; RETURN; 500* Verify that directory and subdirectory exist; DIRECTORY.FOUND=0; CALL PIX.EXISTS(C.IMAGE.PATH:SUBDIRECTORY,DIRECTORY.FOUND); IF NOT(DIRECTORY.FOUND) THEN; COMMAND='COMMAND /C MKDIR ':C.IMAGE.PATH:ITEM<MATCHINVEN.CARRIER>; COMMAND=COMMAND; * For Avgen compiler; PRINT NOTEPRINT:'Creating subdirectory ':C.IMAGE.PATH:SUBDIRECTORY:'...'; CALL PIX.CALL.DOS(COMMAND,1); CTR=0; LOOP; CALL PIX.EXISTS(C.IMAGE.PATH:SUBDIRECTORY,DIRECTORY.FOUND); UNTIL DIRECTORY.FOUND OR CTR>(DIRECTORY.TIME.LIMIT-1) DO; CTR=CTR+1; PRINT BELL:; SLEEP 1; REPEAT; IF NOT(DIRECTORY.FOUND) THEN; DISPLAY.MESSAGE='The directory ':C.IMAGE.PATH:SUBDIRECTORY:' does not exist. Please contact your system administrator further assistance.'; PROBLEM.FOUND=1; RETURN TO 99; END ELSE; PRINT NOTEPRINT:'Directory created !'; SLEEP 1; PRINT NOTEPRINT:NOTE; END; END; C.NEW.IMAGE.PATH=C.IMAGE.PATH:SUBDIRECTORY; RETURN; CASE CHOICER='OVERWRI TE. I MALit'; * Copied in from delete case above !!!; DISPLAY.MESSAGE='Feature currently under development'; OVERWRITE. COMMAND=''; CALL AVSS.WINDOW.EDIT(AV,'AVUP.PROMPT','Enter OVERWRITE to rescan the image for this record ? ','Overwrite WARNING!'; '',OVERWRITE.COMMAND,'9','N','ON'); IF OCONV(OVERWRITE.COMMAND,'MCU')='OVERWRITE' THEN; GOSUB 500; CALL PIX.ERASE.DOS.FILE(C.IMAGE.PATH:SUBDIRECTORY:C.IMAGEID:'.BMP'); PRINT NOTEPRINT:'Original Image Deleted !'; IF MODE='NEW' THEN C.IMAGEID=''; SLEEP 1; NOTE=''; PRINT NOTEPRINT:NOTE; Copied in from Capture case above !!!; IF MODE='NEW, THEN; CURRTIME=TIME(); IF CURRTIME>999 THEN; CURRTIME=CURRTIME[LEN(CURRENTIME)-3,4]; END; C.IMAGEID='T':AV<1,1,1>'R°/3':CURRTIME 'R%4'; END; NOTE='Capturing image ...'; PRINT NOTEPRINT:NOTE; PROGID='C:\SNAPPY\SNAPPT.txt'; PROGID=PROGID; STATE='3'; STATE=STATE; SUCCESSFUL=''; CALL PIX.WIN.RUN(PROGID,STATE,SUCCESSFUL); IF NOT(SUCCESSFUL) THEN; DISPLAY.MESSAGE='TERMiTE could not successfully run the image capturing software. Make sure; running on the desktop and then consult your system administrator.'; PROBLEM.FOUND=1; GOTO 99; END; APP.NAME='Snappy Video Snapshot'; Send

keystrokes to snap picrure..; KEYS='SP'; GOSUB 400; *Sleep while image is paintea on screen; SLEEP 15; * Send carriage return to remove full screen snapshot; APP.NAME='Snappy Snapshot'; KEYS='CR'; GOSUB 400; Send keys to select save; NOTE='Saving image ....'; PRINT NOTEPRINT:NOTE; SLEEP 2; APP.NAME='Snappy - Image1'; KEYS='"S"'; GOSUB 400; Send name of file; SLEEP 2; APP.NAME='Save Snappy Image'; KEYS="":C.IMAGE.PATH:SUBDIRECTORY:C.IMAGEID:'.BMP':"' CR'; GOSUB 4DO; Close Snappy; SLEEP 4; NOTE='Closing Snappy'; PRINT NOTEPRINT:NOTE; APP.NAME='Snappy - ':C.IMAGEID; KEYS='A4'; GOSUB 400; APP.NAME='Snappy - ':C.IMAGEID:'.BMP'; KEYS='A4'; GOSUB 400; *Check to see if Image is really on Disk before flipping rename flag; CODE=0; CALL PIX.EXISTS(C.IMAGE.PATH:SUBDIRECTORY:C.IMAGEID:'.BMP',CODE); IF CODE=1 AND MODE='NEW' THEN; C.RENAME.IMAGE=1; END; RUNTIME=0; TIMEOUT=30; LOOP; SLEEP 1; RUNNING="; CALL PIX.WIN.RUNNING(APP.NAME,RUNNING); RUNTIME=RUNTIME+1; UNTIL NOT(RUNNING) OR RUNTIME>TIMEOUT DO REPEAT; NOTE="; PRINT NOTEPRINT:NOTE; END

Although the invention has been described with reference to the preferred embodiment illustrated in the attached drawing figures, it is noted that equivalents may be employed and substitutions made herein without departing from the scope of the invention as recited in the claims.

What is claimed is:

1. A method for matching and identifying lost inventory from a carrier that has been delivered to a warehouse, the method comprising:

designating at least one piece of inventory as overgoods inventory;

generating overgoods information for each piece of overgoods inventory, including descriptive product information, location information, and information about a carrier associated with the overgoods inventory;

inputting the overgoods information into an overgoods record in a computer;

searching the overgoods records in the computer for records matching preselected criteria, including information about the carrier, to generate at least one matched overgoods record corresponding to matched overgoods inventory; and

notifying the carrier that at least one matched overgoods record matches the preselected criteria.

2. The method of claim 1, further comprising the steps of

identifying a current location of the corresponding piece of matched overgoods inventory from the location information in the matched overgoods record; and

physically relocating the piece of matched overgoods inventory from the current location to a second location for shipment or further processing.

3. The method of claim 2, further comprising the step of printing, with a printer coupled with the computer, a ticket with shipping information for each piece of matched overgoods inventory prior to physically relocating the piece of matched overgoods inventory from the current location.

4. The method of claim 1, further comprising the step of displaying selected overgoods information on a computer screen for each matched overgoods record.

5. The method of claim 4, further comprising the steps of designating at least one matched overgoods record as an on-hold record after generating at least one matched overgoods record and displaying the on-hold designation on the computer screen when selected overgoods information is displayed on the computer screen for the corresponding matched overgoods record.

6. The method of claim 5, further comprising the steps of

de-designating each on-hold record as an on-hold record;

identifying a current location of the corresponding piece of matched overgoods inventory from the location information in the matched overgoods record; and

physically relocating the piece of matched overgoods inventory from the current location to a second location for shipment or further processing.

7. The method of claim 1, further comprising the steps of :

printing a machine-readable overgoods identification label for each piece of overgoods inventory with a printer coupled with the computer, the label having an overgoods inventory code associated with its respective overgoods record; and

placing the overgoods identification labels on their respective piece of overgoods inventory.

8. The method of claim 7, further comprising the step of updating the current location in the location information for each piece of overgoods inventory by inputting the overgoods inventory code into a programmable input device in communication with the computer when the current location changes.

9. The method of claim 1, wherein the step of searching the overgoods records further includes restricting access to selected records based on a log in code supplied by a user of the computer.

10. The method of claim 1, further comprising the step of displaying a selected portion of the matched overgoods record on a display device in communication with the computer.

11. The method of claim 10, wherein the step of displaying a selected portion of the matched overgoods record includes allowing a user of the computer to view the entire matched overgoods record.

12. The method of claim 1, wherein the step of receiving overgoods information further comprises the step of receiving a digital image of the overgoods inventory into the overgoods record.

13. The method of claim 12, further comprising the step of allowing a user of the computer to view the digital image of the matched overgoods inventory associated with the matched record after identification of the matched record.

14. The method of claim 1, further comprising the step of physically relocating each piece of overgoods inventory to an overgoods inventory storage area after designation of the distressed inventory as overgoods inventory.

15. A method for managing overgoods inventory with the aid of a computer, the method comprising:

designating a plurality of pieces of inventory as overgoods inventory;

generating overgoods information for each piece of overgoods inventory, including descriptive product information, location information, and information about a carrier associated with the overgoods inventory;

inputting the overgoods information into an overgoods record in a computer;

searching the overgoods records in the computer for records matching preselected criteria;

identifying records that do not match the preselected criteria and that have been in the computer for a preselected length of time as stale records;

designating the overgoods inventory associated with each stale record as a matured item; and

physically relocating each matured item for release to a salvage process.

16. A method for managing inventory for which delivery to an intended destination has not been made, comprising:

identifying a shipper from which the inventory was acquired;

receiving descriptive product information, including shipper information, for the inventory into records in a computer database;

returning the inventory to the shipper for appropriate disposition;

searching the database for records matching preselected criteria provided by a requesting party corresponding to inventory that did not arrive at its intended destination; and

providing to the requesting party the descriptive product information, including date returned to the shipper, for the inventory records matching the preselected criteria.

17. The method of claim 16, wherein the step of providing information to the requesting party includes providing sufficient information to demonstrate that claims against a carrier of the inventory for loss of the inventory are inappropriate.

18. A method for managing inventory for which consignee information is not known, with the aid of a computer, comprising

identifying a shipper from which the inventory was acquired;

receiving descriptive product information, including shipper information, for the inventory into records in a computer database;

returning the inventory to the shipper for appropriate disposition;

searching the database for records matching preselected criteria provided by the shipper corresponding to inventory that did not arrive at its intended destination; and

providing to the shipper the descriptive product information, including date returned to the shipper, for the inventory records matching the preselected criteria, enabling the shipper to physically identify the inventory corresponding to the matching records.

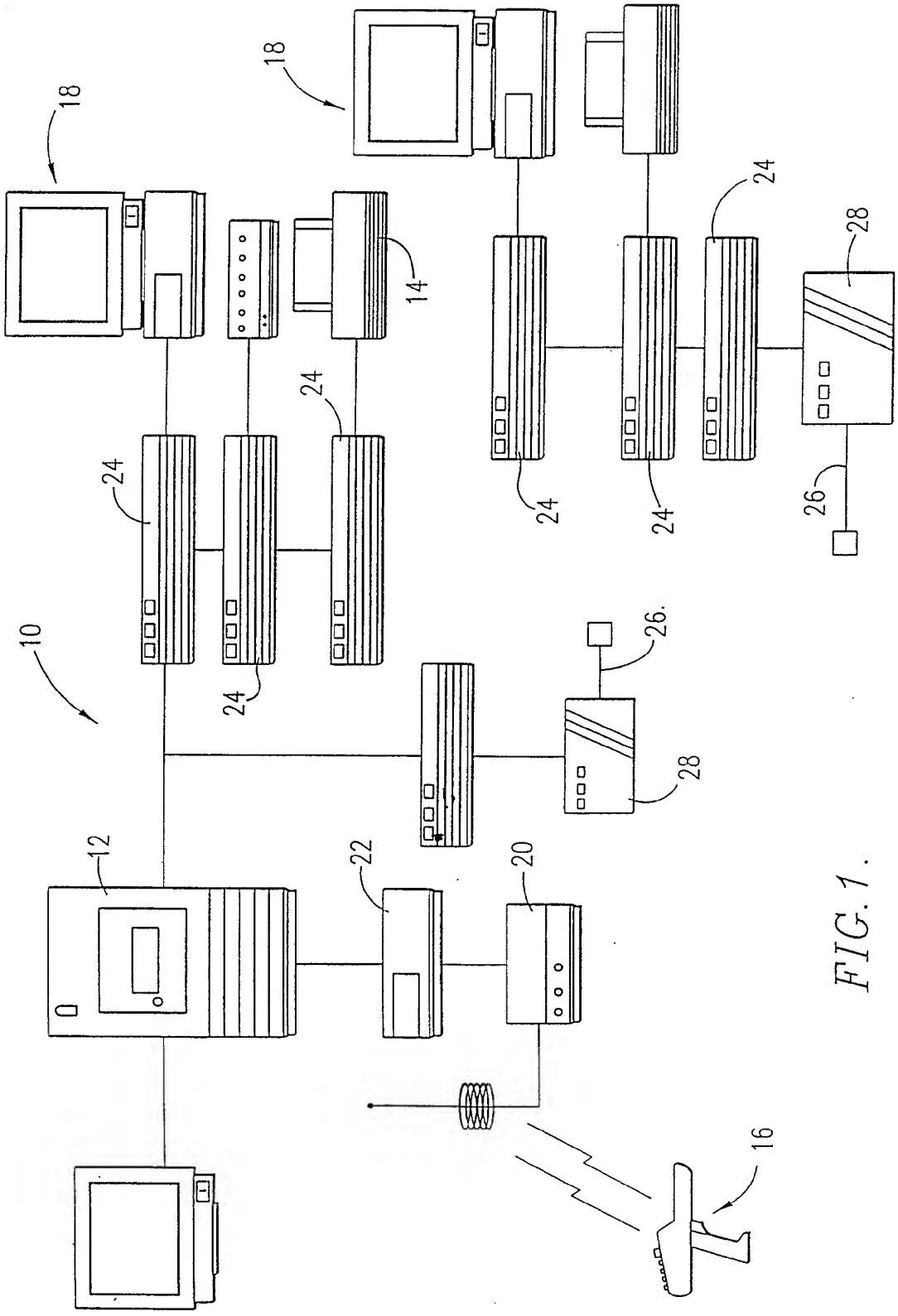


FIG. 1.

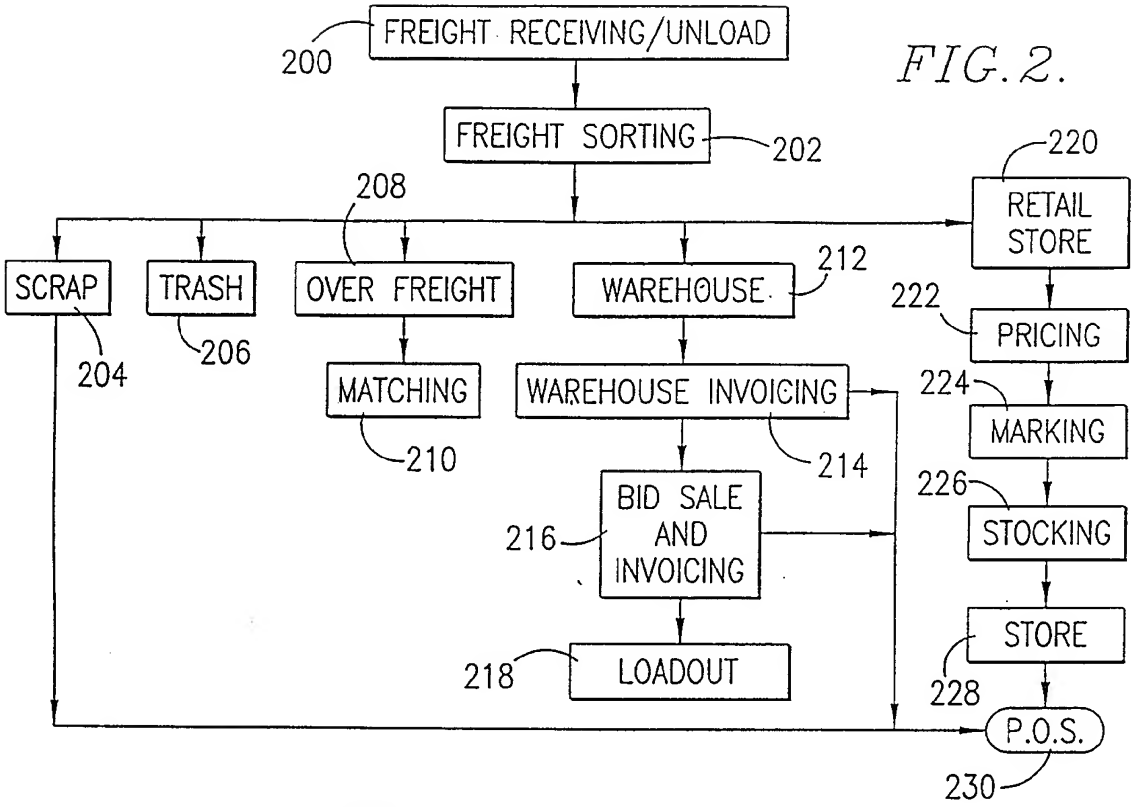

FREIGHT RECEIVING/UNLOAD
200
FREIGHT SORTING
202
208
220
RETAIL STORE
SCRAP
204
TRASH
206
OVER FREIGHT
WAREHOUSE
212
222
PRICING
MATCHING
210
WAREHOUSE INVOICING
214
224
MARKING
BID SALE AND INVOICING
216
226
STOCKING
218
LOADOUT
STORE
228
P.O.S.
230

FIG. 2.

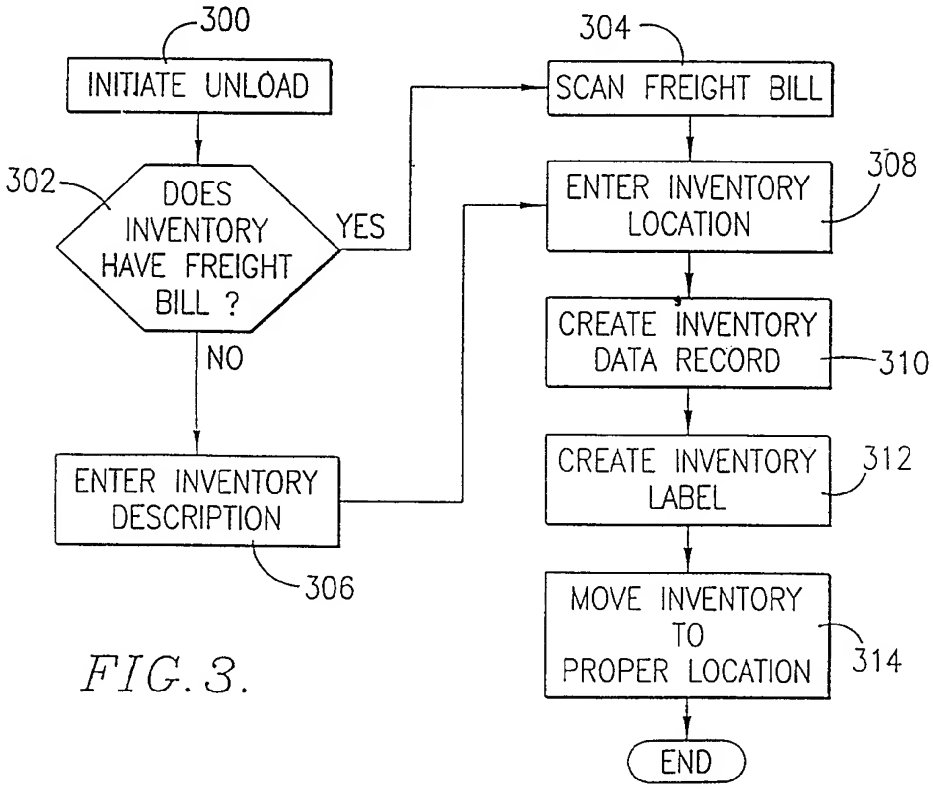

300
INITIATE UNLOAD
304
SCAN FREIGHT BILL
302
DOES INVENTORY HAVE FREIGHT BILL ?
YES
NO
ENTER INVENTORY LOCATION
308
CREATE INVENTORY DATA RECORD
310
ENTER INVENTORY DESCRIPTION
306
CREATE INVENTORY LABEL
312
MOVE INVENTORY TO PROPER LOCATION
314
END

FIG. 3.

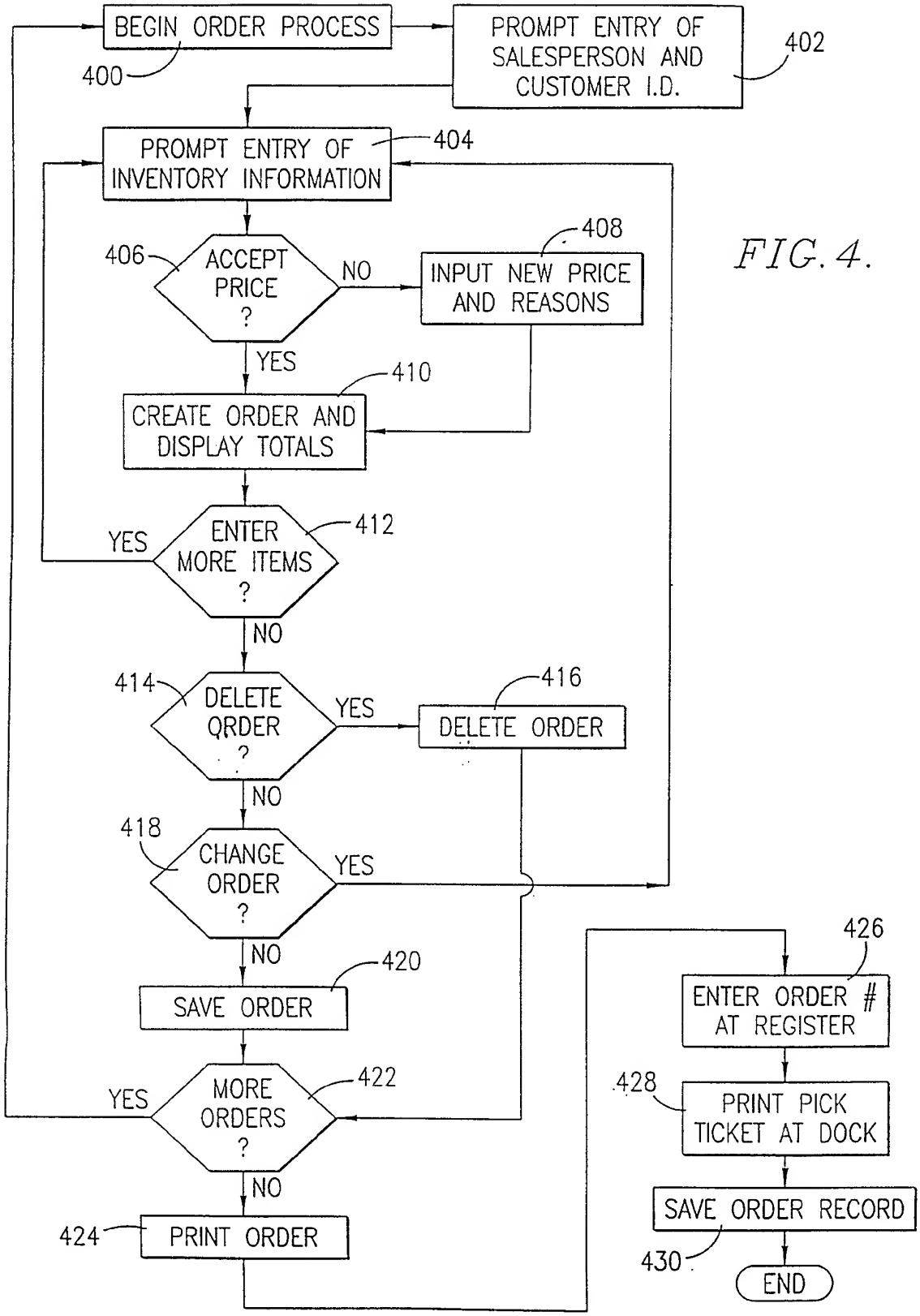

BEGIN ORDER PROCESS
400
PROMPT ENTRY OF SALESPERSON AND CUSTOMER I.D.
402
PROMPT ENTRY OF INVENTORY INFORMATION
404
406
ACCEPT PRICE ?
NO
408
INPUT NEW PRICE AND REASONS
YES
410
CREATE ORDER AND DISPLAY TOTALS
YES
ENTER MORE ITEMS ?
412
NO
414
DELETE ORDER ?
YES
416
DELETE ORDER
NO
418
CHANGE ORDER ?
YES
NO
420
SAVE ORDER
YES
MORE ORDERS ?
422
NO
424
PRINT ORDER
426
ENTER ORDER # AT REGISTER
428
PRINT PICK TICKET AT DOCK
SAVE ORDER RECORD
430
END

FIG. 4.

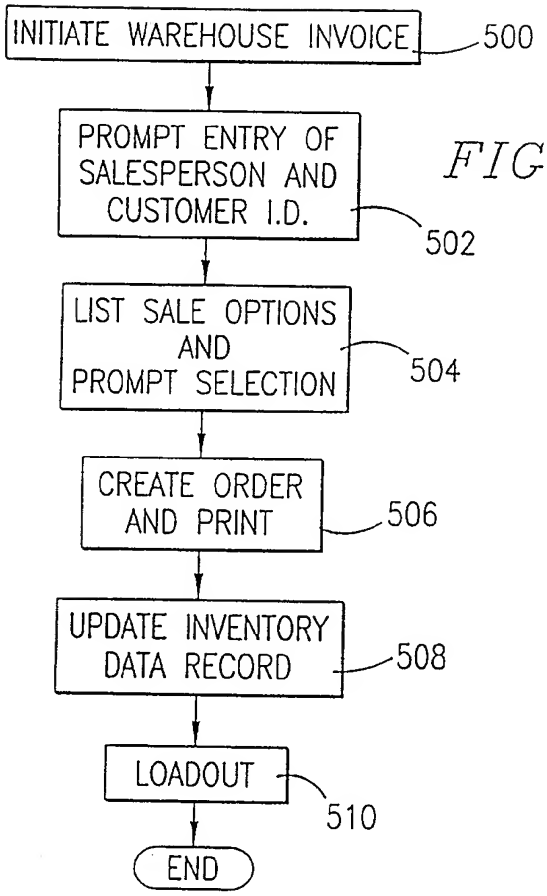


FIG. 5.

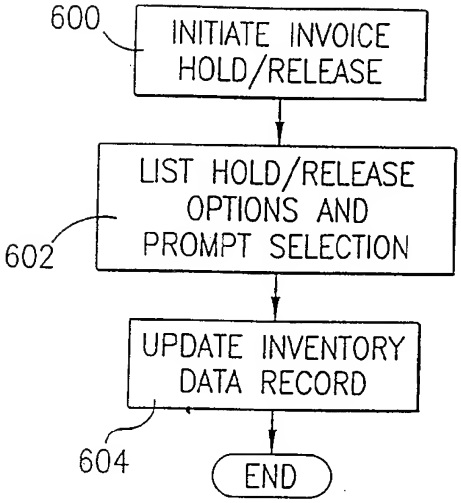


FIG. 6.

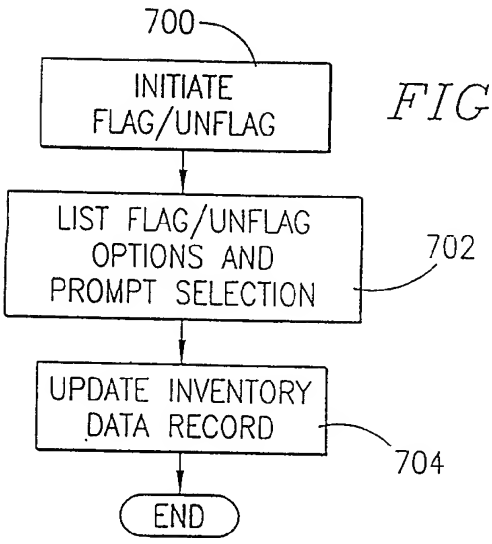


FIG. 7.

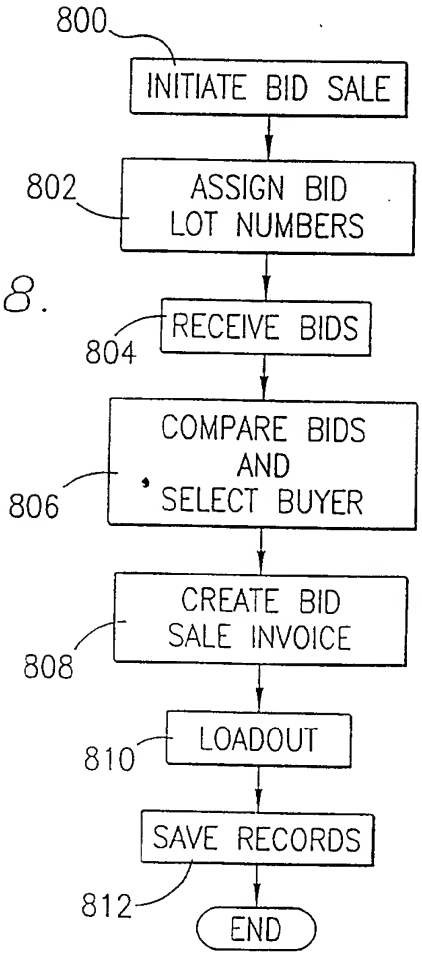


FIG. 8.

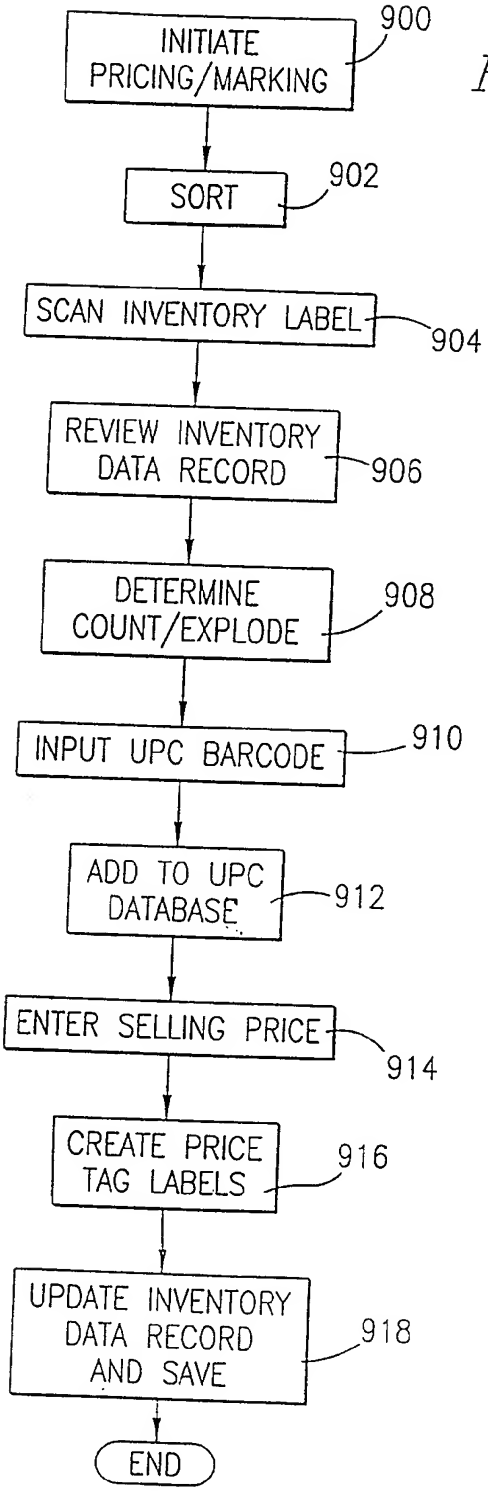
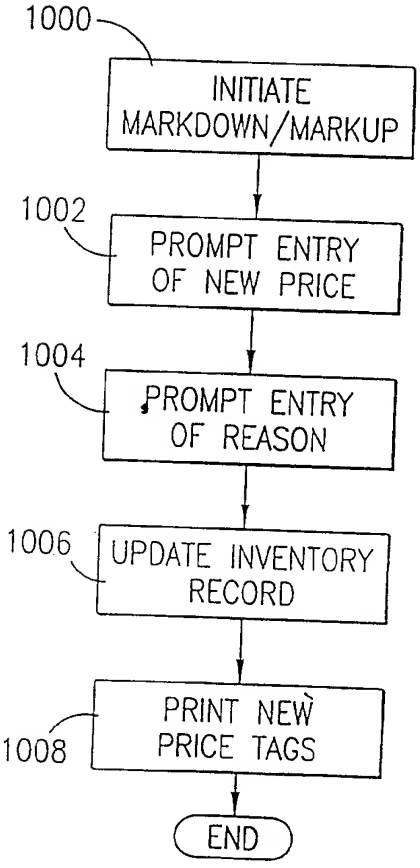

INITIATE PRICING/MARKING
900
SORT
902
SCAN INVENTORY LABEL
904
REVIEW INVENTORY DATA RECORD
906
DETERMINE COUNT/EXPLODE
908
INPUT UPC BARCODE
910
ADD TO UPC DATABASE
912
ENTER SELLING PRICE
914
CREATE PRICE TAG LABELS
916
UPDATE INVENTORY DATA RECORD AND SAVE
918
END

FIG. 9.

FIG. 10.


1000
INITIATE MARKDOWN/MARKUP
1002
PROMPT ENTRY OF NEW PRICE
1004
PROMPT ENTRY OF REASON
1006
UPDATE INVENTORY RECORD
PRINT NEW PRICE TAGS
1008
END

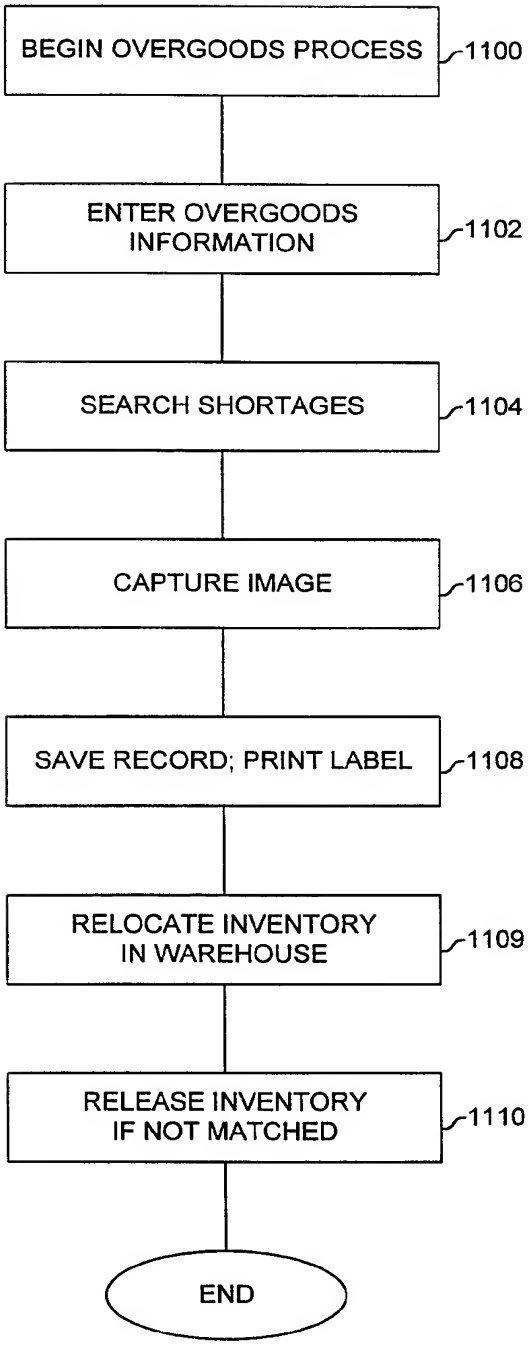

BEGIN OVERGOODS PROCESS
1100
ENTER OVERGOODS INFORMATION
1102
SEARCH SHORTAGES
1104
CAPTURE IMAGE
1106
SAVE RECORD; PRINT LABEL
1108
RELOCATE INVENTORY IN WAREHOUSE
1109
RELEASE INVENTORY IF NOT MATCHED
1110
END

FIG. 11

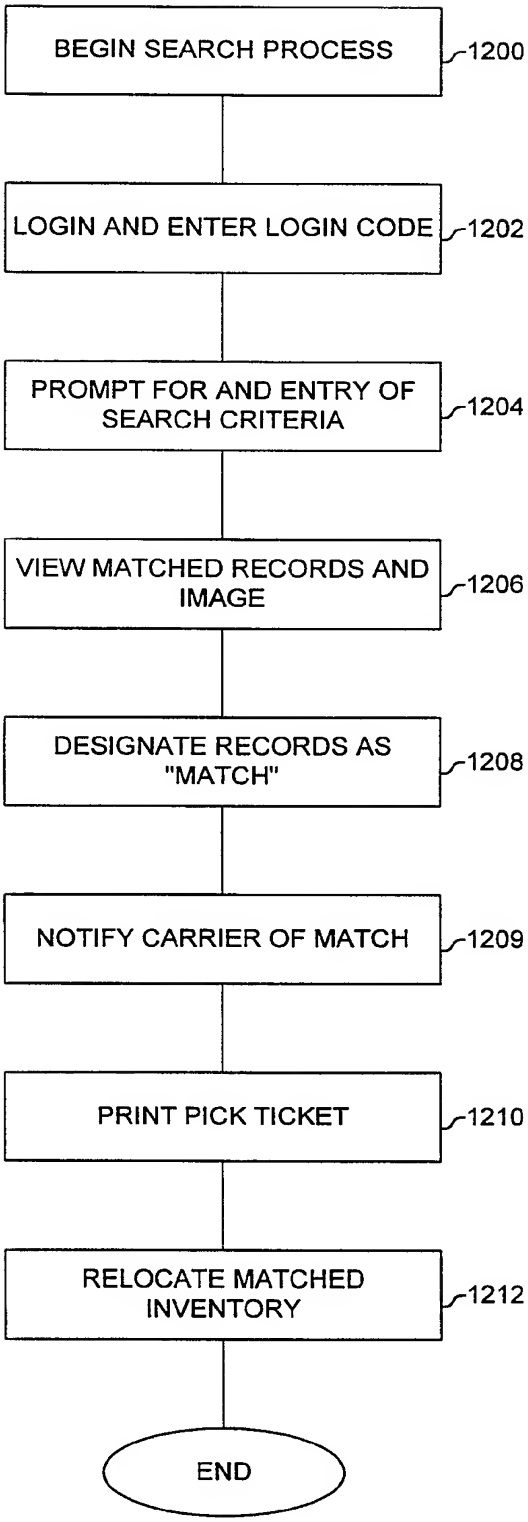

BEGIN SEARCH PROCESS
1200
LOGIN AND ENTER LOGIN CODE
1202
PROMPT FOR AND ENTRY OF SEARCH CRITERIA
1204
VIEW MATCHED RECORDS AND IMAGE
1206
DESIGNATE RECORDS AS "MATCH"
1208
NOTIFY CARRIER OF MATCH
1209
PRINT PICK TICKET
1210
RELOCATE MATCHED INVENTORY
1212
END

FIG. 12

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/19821

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 17/60
US CL : 705/28

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 705/22, 28; 364/479.06;

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS including JPO and EPO

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 4,074,120 A (ALLRED et al) 14 February 1978	1-18
A	US 4,558,318 A (KATZ et al) 10 December 1985	1-18
A	US 4,336,589 A (SMITH et al) 22 June 1982	1-18
A	US 4,340,810 A (GLASS) 20 July 1982	1-18
A	US 4,525,071 A (HOROWITZ et al.) 25 June 1985	1-18
A	US 4,887,208 A (SCHNEIDER et al) 12 December 1989	1-18
A	US 5,319,544 A (SCHMERER et al) 07 JUNE 1994	1-18
A	US 5,334,822 A (SANDFORD) 02 August 1994	1-18

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search
29 SEPTEMBER 1999

Date of mailing of the international search report
08 NOV 1999

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer
ALLEN MACDONALD
Telephone No. (703) 308-0000

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/19821

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,631,827 A (NICHOLLS et al) 20 May 1997	1-18
A	US 5,646,389 A (BRAVMAN et al) 08 July 1997	1-18